GROUP DECISION MAKING WITH FEEDBACK

Amnon Tamir

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

GROUP DECISION MAKING WITH FEEDBACK

by

Amnon Tamir

September 1979

Thesis Advisor: F.R. Richards

Approved for public release; distribution unlimited.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Group Decision Making with Feedback		5. TYPE OF REPORT & PERIOD COVERED Master's Thesis; September 1979
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Amnon Tamir		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		12. REPORT DATE September 1979
		13. NUMBER OF PAGES 119
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

A computer tool is developed for the purpose of eliciting group utilities for multiple attributes of one complex system relative to a base line. The procedure accommodates multiple users simultaneously providing anonymous

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601 |
(Page 1)

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

(20. ABSTRACT Continued)

feedback to each user to aid in the process of assessing utilities.

The procedure provides complete visibility to a manager (umpire) of changes to the data base, so that the process can be monitored in real time. The software is written so that it is completely self-documented and user friendly.

Approved for public release; distribution unlimited.

Group Decision Making with Feedback

by

Amnon Tamir
Major, Israeli Air Force
B.S.A.E., Technion, Israel Institute of Technology, 1970

Submitted in partial fulfillment of the
requirements for the degrees of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

and

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL

September 1979

ABSTRACT

A computer tool is developed for the purpose of eliciting group utilities for multiple attributes of one complex system relative to a base line. The procedure accommodates multiple users simultaneously providing anonymous feedback to each user to aid in the process of assessing utilities.

The procedure provides complete visibility to a manager (umpire) of changes to the data base, so that the process can be monitored in real time. The software is written so that it is completely self-documented and user friendly.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS

The list of people whom I "bugged" while I worked on this thesis is way too long to be wholly reproduced. However, I would like to expressly thank Professor Larry Kou with whom I started on the Computer Science aspect of the thesis, Also, I would like to thank Professor Gordon Bradley, who took Professor Kou's place when he left, Professor Russell Richards who did more than his share and is more than an advisor to me, Dr. Cynthia Irwine, who led me in the mysteries of UNIX and "C", and last but not least my dear wife Ariane whose patience was an invaluable asset to my work.

I. INTRODUCTION

Utility models are often used for the purpose of evaluating complicated systems or to select among competing alternatives. The utility assessment process usually determines a collection of system attributes which are used collectively as surrogates for the system. A decision maker (or group of decision makers) is then asked to evaluate the utility of each attribute of the system. These unidimensional utilities are then combined into a multiattribute utility measure of the entire system using some sort of rational aggregation procedure which depends on the properties of the unidimensional utilties.

Multiple decision makers are frequently used for determining the unidimensional utilities for the individual attributes. This is because it is rare to find one person who is expert on all attributes. Perhaps nowhere is the old adage "two heads are better than one" more true than in utility assessment. In this thesis a computer tool is developed to aid in the process of obtaining the unidimensional utilities.

Background information about the utility problem is provided in Chapter II. The decision making problem is discussed; multiattribute utility is introduced; and the group decision making with feedback problem is examined. A case is made for the need for an automated tool for extracting group utilities.

Chapter III discusses the data and software requirements for a computer tool used to help subjects determine utilities for each of several attributes. Desirable interactive concepts of such a tool are described. Hardware requirements for the computer and input/output terminals are also discussed.

Chapter IV describes the various user programs that have been written to interact with the subjects to obtain utilities. It also describes the programs that are available to the monitor (umpire) to allow him to watch over the process and to keep track of the status of each user. Also included in Chapter IV are descriptions of various utility programs that were written to guarantee data base integrity and to aid in the analysis of the utility data.

Chapter V provides a user's manual with a sample terminal session as an example of the use of the tool. The user's manual should suffice for documentation to be provdided to a user as to what he is required to do to utilize the antomated procedure.

Finally, in Chapter VI, we discuss present limitations of the procedure in terms of the number of users, the number of attributes, total core and the like. We also describe possible future extensions of the process to allow for enhanced graphical output, and we discuss other applications of the tool outside the area of utility assessment that the procedure can be used for with only minor changes.

II. BACKGROUND

A. SUBJECTIVE EVALUATION OF ALTERNATIVES

Consider the problem of deciding among several possible alternatives which we label as $A_1, A_2, \ldots, A_n$. Each alternative has some value or utility to us which depends on the state of nature which is outside our control. Let the possible states of nature be denoted by $S_1, S_2, \ldots, S_k$. For each pair (S_i, A_j) there is a result r_{ij} (see figure 1). The collection of results are what have value or utility to the decision maker(s) (see figure 2).

Decision theory is concerned with how decision makers should select among competing alternatives in such a framework. The theory considers as separate cases decision making under uncertainty and decision making under risk. In the former, the probabilities for the different states of nature are assumed known; in the latter the probabilities are unknown. In both cases, however, the decision maker(s) is required to assess the utility u_{ij} of each result r_{ij}. The utility assignments are subjective. We assume that they have been made rationally in accordance with the set of axioms of von Neumann and Morgenstern [Ref. 1].

As an example of this decision framework consider a problem of selecting among two available aircraft and one in development for the purpose of providing close air support for a mission planned against enemy armored forces. The three alternatives are:

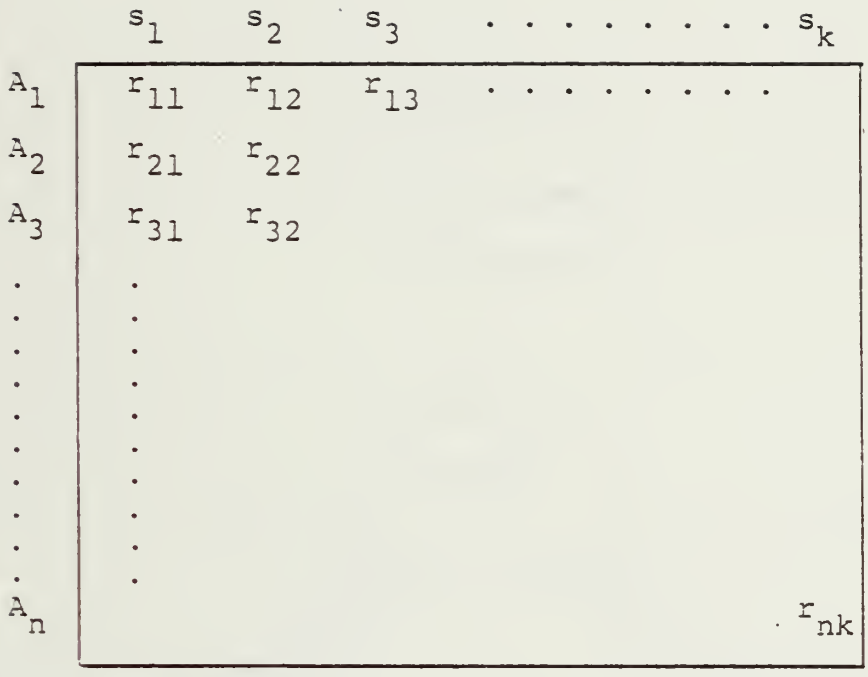


Figure 1

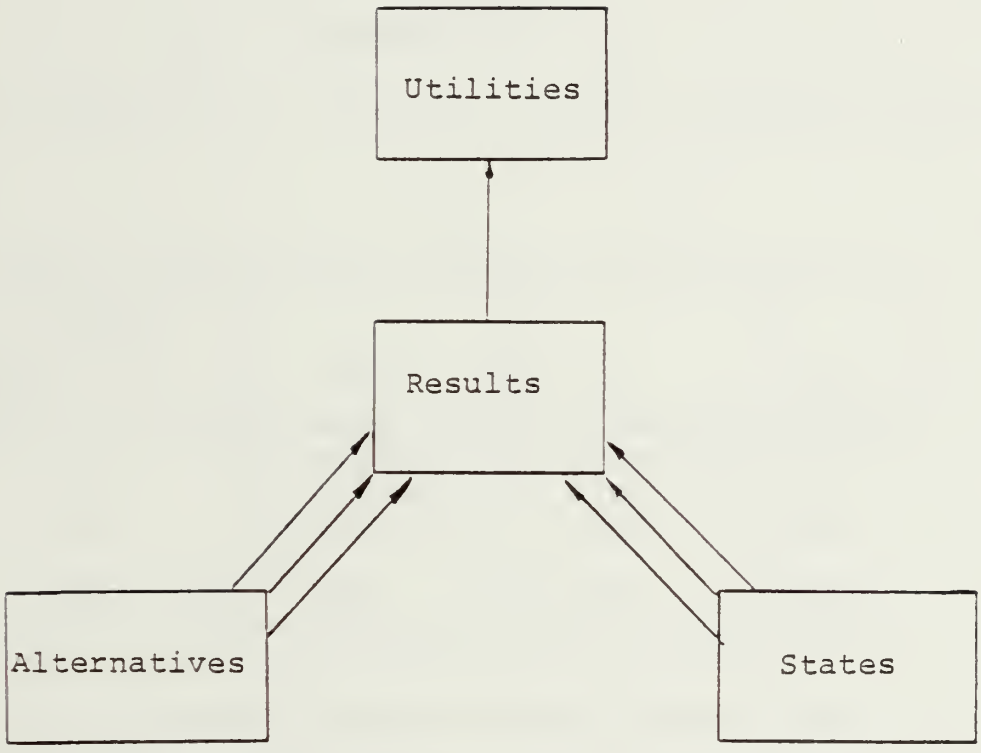


Figure 2

A_1 = A-4

A_2 = F-4

A_3 = A-X

The states of nature that may affect the results of the mission are the visibility at the time of the mission.

Let us consider the states

S_1 = clear day

S_2 = clear night

S_3 = cloudy day

Now suppose that the result matrix is given as follows:

r_{ij}

Alternatives \ states of nature	S_1	S_2	S_3
A_1	acceptable	acceptable	bad
A_2	very good	good	bad
A_3	good	good	good

Note in the table above that the results need not be quantitative. Finally, suppose that the decision maker(s) has assessed the following utilities for the results shown in the table:

U_{ij}

states of nature / Alternatives	S_1	S_2	S_3
A_1	3	3	-2
A_2	10	5	-2
A_3	5	5	5

There are many different procedures for selecting among the alternatives after the alternatives and the states have been identified and the utilities assessed. No attempt will be made to discuss the different procedures. For a good discussion see Keeny and Raiff [Ref. 2], Zimmermann [Ref. 3], Raiffa [Ref. 4] and Fishburn [Ref. 5]. In this thesis we are concerned primarily with the problem of eliciting the utilities from the decision maker(s).

B. MULTIATTRIBUTE UTILITY MODEL

The utility measurement of complicated systems is one of the most difficult problems facing decision makers. The measurement of utility must take into account all the anticipated uses and conditions for which the system would be utilized.

One approach to assessing utilities of complex alternatives is to ask decision makers to assess the utility of the alternatives directly, without explicitly determining utilities of the individual attributes. This is what most of us

do in our minds when making everyday decisions. The decision makers may think hard and seriously about the alternatives and attributes of the alternatives, but no formal model is developed, and no formal analysis is done. The approach places quite a burden on the decision maker forcing him to integrate in his mind many different types of information such as the scenarios (states of nature), the attributes, the importance of the attributes and tradeoffs. This approach may lead to good decisions but it is generally unsatisfactory because it provides no "audit trail" for justifying its conclusions to others. Since the model is not specified, no scrutiny can be made of the assumptions or factors that led to the conclusion. Another disadvantage is that the approach does not allow for sensitivity analysis or discovery of which attributes or characteristics of the alternative systems are most important.

A second approach is to identify the key attributes of the alternative systems that have value to the decision makers. Let $x_1, x_2, \ldots, x_r$ represent r such attributes, and let $x_i = X_i(A)$ represent the level of attribute X_i possessed by alternative A. These levels are used collectively as a surrogate for the alternative systems. Single dimensional utilities $u_i(x_i)$ are then assessed for the levels of the attributes. This approach thus allows a decision maker to focus on each attribute one at a time. If care is taken in deriving the list of attributes so that the set "captures the essence" of the alternatives, then this approach minimizes

the likelihood that important considerations "fall through the cracks". It also insures that the final overall assessments will not be influenced unjustifiably by placing too much importance on only a few of the attributes of the system. The single dimensional utilities are then aggregated into an overall system utility as follows:

$$U(A) = U(x_1, x_2, \ldots, x_r) = f(u_1(x_1), \ldots, u_r(x_r))$$

There is quite a bit of recent literature on the form that the aggregating function f should take. See especially Keeny and Raiffa [Ref. 2]. One form of f that is commonly used is

$$f(U_1(x_1), \ldots, U_r(x_r)) = \sum_{i=1}^{r} W_i U_i(x_i) \quad .$$

This form can be shown to be valid when certain types of independence conditions over preferences for the attributes are satisfied.

The actual form of the utility aggregation function is not of concern in this thesis. Instead, this thesis concentrates on the task of finding the individual attribute utilities $U_i(x_i)$.

C. GROUP DECISION MAKING WITH FEEDBACK

A single decision maker is rarely qualified to evaluate and assess the utility of each attribute of a complicated

system. One would probably be better off getting a group of experts together to share their experience and assess the utilities as a group. Getting a single number out of the information supplied by a group is a problem by itself. In reference [6] a methodology for group decision making with feedback is proposed.

The methodology suggests a procedure that has three basic steps:

1. Utility assessment.
2. Information aggregation.
3. Feedback.

In the first step, the group members assess utilities for each attribute of the system. The procedure administrator then aggregates all the information about each attribute and feeds information back to the group members. The group members examine the responses of their colleagues and modify their own responses as they desire. See figure 3.

Because utility values are not unique (they are unique up to a linear transformation) and because relative utilities and values are frequently easier to evaluate than are absolute values, we have chosen to collect the required information in terms of value ratios with one system selected as a baseline for comparison. With the ratio scale (fig. 4) a group member is required to provide two numbers as data for each attribute.

1. The first is the ratio of the value of the new system to the value of baseline system. For example: if the range of the new system is 150 NM and the range of the baseline

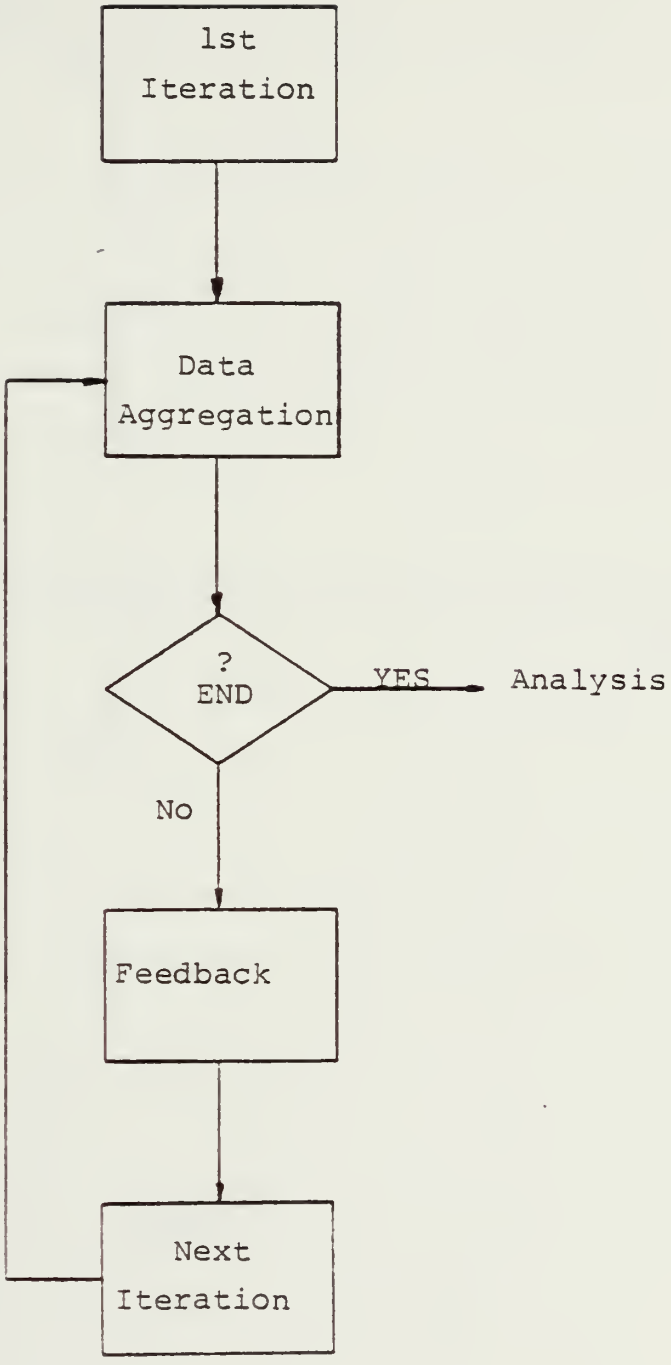


Figure 3

The Group Decision Making With Feedback Process

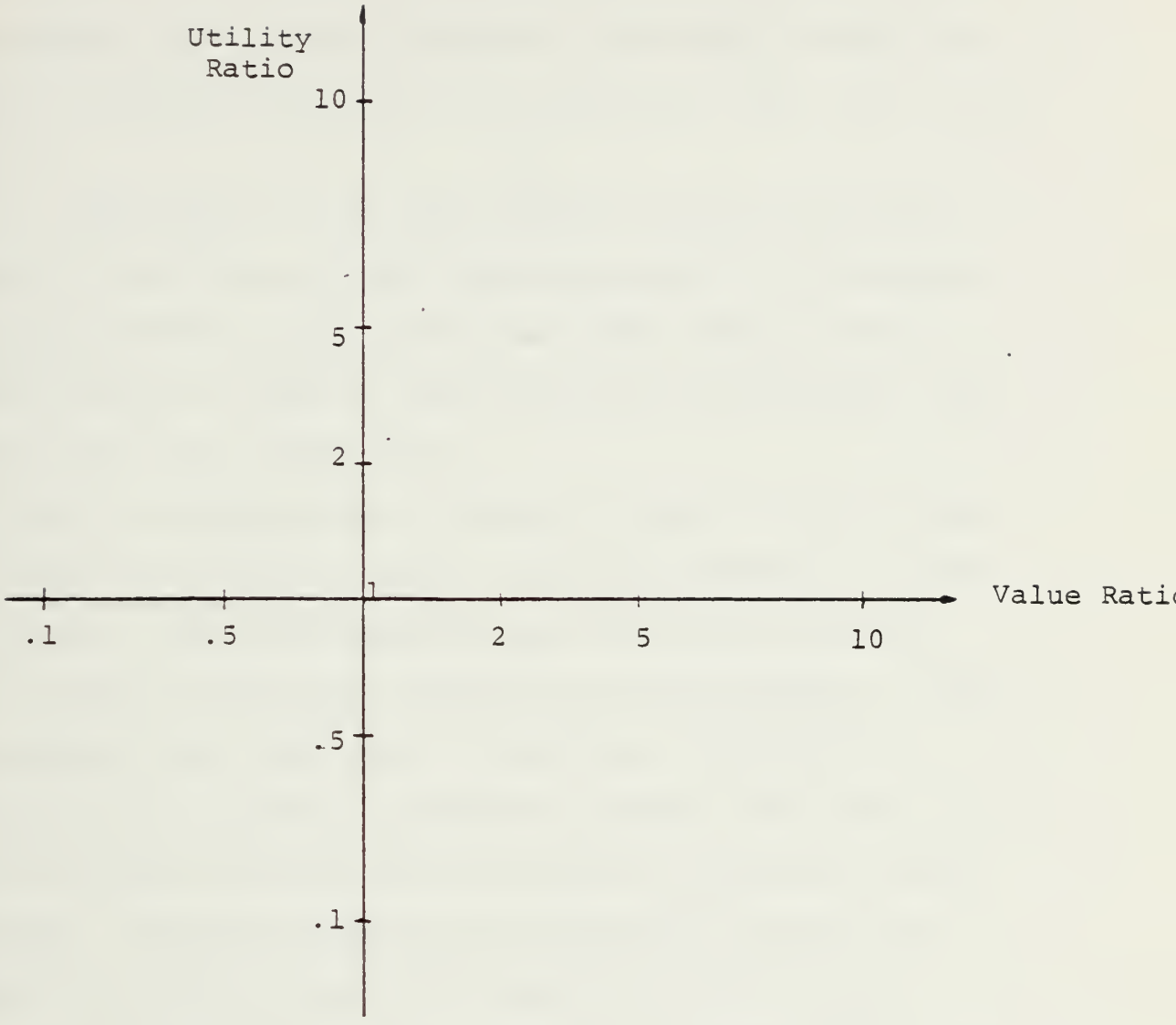


Figure 4

radar is 100 NM the value ratio is 1.5. The ratio is also convenient for handling attributes which are non-measurable. For example, if a subject thinks the new radar is 30% more reliable than the baseline, the reliability value ratio would be 1.3

2. The second is the ratio os the utility of the new system to the utility of the baseline system. For example: if having 50% more range on the new radar makes it have a utility twice as great as the utility for the baseline, the utility ratio for range is 2.0.

Since a single person is rarely an expert in all of the attributes of a complex system, the group members are asked to provide a self-proficiency rating for each attribute. This is done in the first iteration of the procedure. After all subjects have submitted an attribute, the data are summarized and fed back in graphical output. The graph for a given attribute contains as many points as there are group members. The group members are encouraged to examine the feedback and modify their last inputs if they so desire.

Once the data are in a database on a computer, the wide variety of statistical tools can be used to evaluate and aggregate the final answers into single utility numbers for all attributes, which may then be combined into an overall system utility value.

Administering this procedure manually is very time consuming and awkward. An iterative computer program is needed to

handle the data collection and feedback process. This thesis describes such an interactive computer program.

III. SOFTWARE AND INPUT/OUTPUT REQUIREMENTS

A. DATA CHARACTERISTICS

Essentially two types of data are required. The first type includes data which are not affected by feedback and which will likely be entered only once by each user. This includes the user profile and information about the user's self-proficiency evaluation on each attribute and quadrant selection. The second type is more dynamic, being directly affected by feedback information on the evaluations of the cohorts of the user. This data consists of the pair of numbers, value rates and utility ratio, for each attribute. The user may update these values as often as he desires.

The analysis that will be performed on the input data requires that the entire data input process be reconstructed. The analyst needs to know the entire sequence of inputs for each user for each attribute. Furthermore, he needs to trace through the input chronologically so that he can tell what feedback may have influenced specific inputs. This requires that the software insert a clock time with each data entry. Since the data analysis will be performed on the Naval Postgraduate School IBM 360/67 computer to take advantage of the powerful data analysis software already available in APL, the data will have to be transportable in a format acceptable to the IBM computer.

B. SOFTWARE CONSIDERATIONS

Many considerations influenced the design of the software. Most important is the requirement that each user simultaneously access the data base interactively. Also since the users will be available for only a short amount of time , extensive training in the use of the software will not be feasible. Thus the programs must be self-explantory containing internal documentation and they must be user friendly assuming no prior computer training. The software should be written to protect the identity of each user from the other users to prevent intimidation, but should allow the project administrator to monitor the performance of each user in real time. This is to allow the administrator to observe the progress of each user. The administrator can thus detect problems that the users may be having and communicate instructions to selected users during a session. Finally, to be useful as a general tool for the assessment of group utilities of several attributes of one system relative to a specified baseline, the programs should be flexible to allow for changes in the systems being evaluated without major software changes.

C. INTERACTIVE CONCEPTS

It is difficult to provide a friendly interface between a sophisticated unforgiving machine and a user assumed to be untrained in the use of computers. The burden of accomplishing the interface falls on the software. Thus the software must

incorporate many human factors considerations. The information displayed to the user should be simple, clear and concise so that the user can grasp the essential information quickly. The time delay from keyboard entry to the terminal response should be no more than a few seconds. The software should provide the user an option with regard to the amount of detail contained in the instructions printed at the user's terminal. As users become more experienced during a terminal session, he will likely grow weary of repeatedly reading the same detailed instructions. He should therefore be allowed to reduce the verbosity of instructions. In addition to allowing the user to reduce verbosity, the software should also allow the user to request additional information or instructions. A "help" feature should be built into the software to allow the user to obtain online documentation at any time during a session without disturbing the flow of the program. Finally, the software should be built to provide protection of the data base from either the malicious intent of a user or from accidental or erroneous responses.

D. INPUT/OUTPUT TERMINALS

Because of costs, portability, and response requirements a "dumb" video CRT terminal with a 1200 baud-rate transmission capability, standard alphanumeric keyboard entry, 23 lines per screen, and 80 characters per line was selected as the basis input/output device around which the software would be designed. Many different terminals satisfy the above

requirements providing ready availability for testing and implementation at the Naval Postgraduate School. This choice of terminals allows the user to view only a single quadrant (which he selects) containing the value ratios and the utility ratios for a selected attribute since the entire graph cannot be displayed with adequate resolution. Fortunately, this does not present serious problems since one would expect most or all users to input data into the same quadrant for a given attribute. In order to provide each user access all the values for a given attribute the software should notify each user of the distribution of data over the four quadrants and allow each user to display any selected quadrant.

Other more expensive types of intelligent terminals could certainly enhance the application of the process. Color and graphics capability would allow the users to distinguish among the data according to the self-proficiency rating of the respondents and would allow greater resolution. Also a split screen capability would allow for some information, such as the attribute list to be maintained on a portion of the screen while other parts are updated as necessary. This would reduce the input/output response time while providing more informative displays. Even though the software is designed around the "dumb" video terminal, it should be flexible enough to allow for easy modification for a more sophisticated terminal.

IV. SOFTWARE DESCRIPTION

A. THE USER'S PROGRAM

1. External Declarations

Most of the variables are declared externally so that they can be used globally.

The maximum number of users and attributes and the size of the basic data record is defined so as to make it easy to change the system capacity.

The basic structures that constitutes the data bank are defined and character arrays are initialized.

2. Main ( )

The main program handles the general flow of the process. It performs the linking, opening and unlinking of the various files needed.

A unique user number is associated with each user and the basic loop of the program is entered.

3. Intro ( )

The introduction routine explains the "rules of the game" and describes the basic procedure that the user will follow.

Intro ( ) gives examples of use and sample displays. The user is allowed to page back and forth through the introductory instructions until he feels that he understands what he is required to do and what options are available to him.

The user can re-enter the Intro ( ) subroutine at almost any time he wants by typing "I" (see subroutine attrib ( )).

4. Prep ( )

The preparation routine performs the first iteration and asks the user for two basic pieces of information regarding each attribute:

(1) The user's self-rated proficiency with respect to an attribute on a scale of 1 (low) to 5 (high).

(2) The quadrant in which he will enter utility information. Once the user enters his proficiency it may not be changed at a later time. However the quadrants may be updated later at any time.

After the entry of the proficiency and the quadrant information, the chosen quadrant is displayed for the given attribute and the user is prompted to enter his first evaluation of the value and utility ratios. This sequence is repeated for all the attributes.

5. Menu ( )

This menu routine displays to the user the list of attributes and a number by which the attribute will be referenced. In each attribute there will also be a column called "changes", which gives each user a count of the number of times that other users have updated the information for each attribute, since he last made a utility entry for the attribute.

6. Attrib ( )

This routine is the main working routine. It starts by displaying to the user the current utility data for the selected attribute and then prompts each user for a response. The user has 7 available options which he can type.

"y" - indicates that the user wants to enter new utility data. The user is then prompted for the value and utility ratios.

"n" - indicates that no new data is to be entered. The program returns to main ( ), to prompt for a new attribute.

"a" - indicates that the user wants to look at the attribute list. The menu ( ) is displayed.

"q" - indicates that he wants a display of the utility data for a different quadrant.

"I" - calls up the introduction routine.

"E" - indicates that the user wishes to terminate the session.

"h" - or any other character not within this list of legal options. A list of the options is displayed.

7. Outdata ( )

The outdata routine generates a CRT display of the most recent data entered by all the users in any selected attribute. The data is displayed to each user in the quadrant which he selects.

A fresh copy of the data is read from the file system and the attribute value ratio and the utility ratio are extracted. The display quadrant is determined and the proper display subroutines are called.

8. Q_1, Q_2, Q_3, Q_4

Four quadrant subroutines, one for each quadrant translate the attribute value and the utility ratios into the proper scale for display. The routines will prepare a character array to be displayed. If more than one point of data falls in one cell, the number of points is displayed as a digit. Otherwise, the blank is displayed.

9. Graph 1 ( ) - Graph 4 ( )

Four graph subroutines, one per quadrant, display the proper axes and the data points in that quadrant. Each subroutine also displays a small four-quadrant figure showing the number of current entries in each quadrant for the selected attribute.

10. Indata ( )

The indata subroutine creates the new data records and writes them into the proper files in the file system. Before writing, the subroutine will look for permission to write into the data base. As soon as permission is given, the information is written into the files.

B. THE MONITOR'S PROGRAMS

1. Clear

The program "clear" initializes the data base files. A warning is given to the monitor that the program will destroy any existing data in the file. The monitor can quit and save the file contents elsewhere.

2. Atlst

The program "atlst" enables the monitor to create or update an attribute list at any time.

3. Tbox

The program "tbox" is a small data base administrator, created to maintain data base integrity. It prevents the user's processes from overwriting information in the data bank. The program will accept requests for write permission into the data files from all the users and will issue the write permission to only one user at a time.

4. An

The program "an" enables the monitor to monitor the progress of the working session.

The program permits the monitor to do the following:

a. Data records in numerical presentation can be displayed for any single user or all users and any single attribute or all attributes.

b. The data in graphical presentation (as the user sees it), can be displayed for any user and any attribute, as they are at any selected time, present or past.

c. The data in graphical presentation (as the user sees it), can be displayed for any user and any attribute, as they are at any selected time, present or past - but only for users from a chosen proficiency and up.

The program runs continuously until terminated by the monitor.

5. Boxstop

This little program will send a message to the administrator tbox to stop execution orderly.

6. L1

The program "l1" takes an entire data base and transforms all the integers into long integers, so they will be compatible with the IBM/360 format.

7. Tape

The command file "tape" loads the output of the program "l1" on a tape for transportation to other computers for analysis of the data.

8. Tapdsk, Extpdsk

The FORTRAN program "tapdsk" reads the data from a tape and stores it into CMS files.

The program "EXTPDSK" is an EXEC routine under the cp/cms time sharing system running on the IBM 360/67 at NPGS. The program defines the proper CMS files for storage of the data bank taken from the PDP-11 and runs the program "tapdsk".

V . USERS'S MANUAL AND SAMPLE SESSION

Each memeber of the utility assessment team will be given a preliminary briefing about his role and about the assessment and feedback process. After the briefing each user will be provided a login name, which will uniquely identify the user, and a password that will provide him access to the system. He should then switch on his video terminal. The UNIX operating system will then prompt the user for his login name and the password. After a successful login a daily message will be displayed followed by the single character '%' which is the UNIX prompt symbol. The user should then type the program name 'qdm' (without the quotes). This causes the users's program to begin execution, and the introductory phase is entered. The following thirteen pages show the information displayed in the introduction.

GROUP DECISION MAKING WITH FEEDBACK

This program will help you as a member of the utility assessment group
assign utilities to the attributes of the specified systems.

We begin with an introduction which will provide instructions on
the use of the program.
A page of information will be displayed. After you read and understand
the page you must respond by typing one of the following
characters and then a 'return' :

n - to see the next page.

p - to see the previous page.

j - to jump over the remaining introduction.

Now enter your request :

FIRST ITERATION :

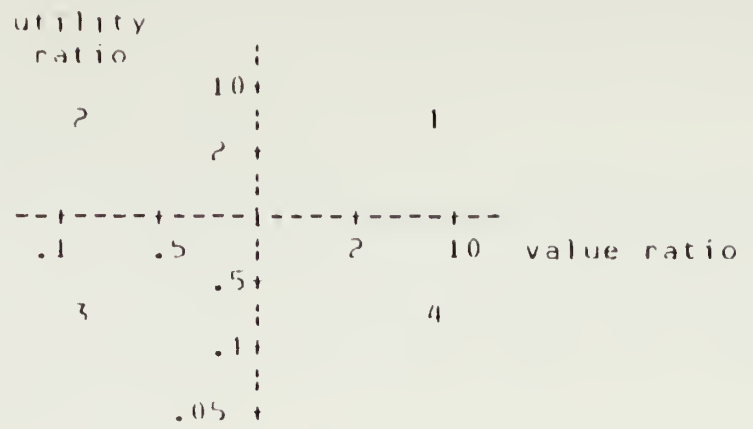
During the first iteration you will be asked to rate your own proficiency for evaluating the utility of each attribute by selecting an integer from 1 (low) to 5 (high).

```
    1    2    3    4    5
    +----+----+----+----+    SELF-PROFICIENCY
  low                 high  RATING
```

You will not be able to change your self-proficiency rating for any attribute after you have entered it initially. Thus be careful in entering the data.

During the utility assessment proces you will be shown a graph of utility ratio versus value ratio for each attribute. Because of screen size and resolution limitation all four quadrants cannot be displayed. Therefore you will be asked to select one of the four quadrants to be displayed for each attribute.

Enter request (n/p or j) :



For each attribute you will be shown the single quadrant which you select. Unlike the self-proficiency rating which cannot be changed you may, at any time, change the quadrant which is displayed for an attribute.

Enter request (n,p or j) :

Comments about the Quadrants :

We expect most of you to agree on the quadrant which contain the utility data. If you think that large value ratios are preferred then you should select either quadrant 1 or quadrant 3. Quadrant 1 would be selected if you think that the subject system has more of the attribute than the baseline system (i.e. value ratio >= 1); quadrant 3 if value ratio is < 1. If you think smaller value ratios are preferred then select either quadrant 2 or 4.

Enter request (n,p or j) :

EXAMPLE :

The situation is transportation for a salesman who travels 40,000 miles per year by automobile over mostly interstate highways.
The baseline system is the current midsize fleet vehicle owned by the parent company. The alternative system under consideration is a new vehicle produced by another auto manufacturer.
The attribute under consideration is COMFORT. Certainly, everything else considered equal, more comfort is preferred. Therefore you should select quadrant 1 if you think that the alternative car is more comfortable then the baseline. Select quadrant 3 if you think the baseline is more comfortable.

Enter request (n,p or j) :

After you enter your self-proficiency rating and the quadrant number for an attribute, the selected quadrant will be displayed, and you will be asked to enter your assessment of both the value ratio and the utility ratio for the attribute in the given scenario.

Enter request (n,p or j) :

FEEDBACK PHASE :

After you have entered initial values for each of the attributes, you will be allowed to display the data input by all of the team members for any selected attribute. Anonymity will be maintained.
You will be able to see all of the entries, but you will not know who entered a given data point.
You will be prompted to select an attribute with the following statement:

Enter the # of the att. you want. Enter 0 to get the att. list.

If you type a number between 1 and n (n is the number of attributes) you will obtain a graphical display of the most recent data from all team members for the selected attribute.

If you enter 0 you will obtain the attribute list with each attribute name, an identifying number, and a tally of the number of updates that have transpired since you last entered data. See the example on the next page.

Enter request (n,p or j) :

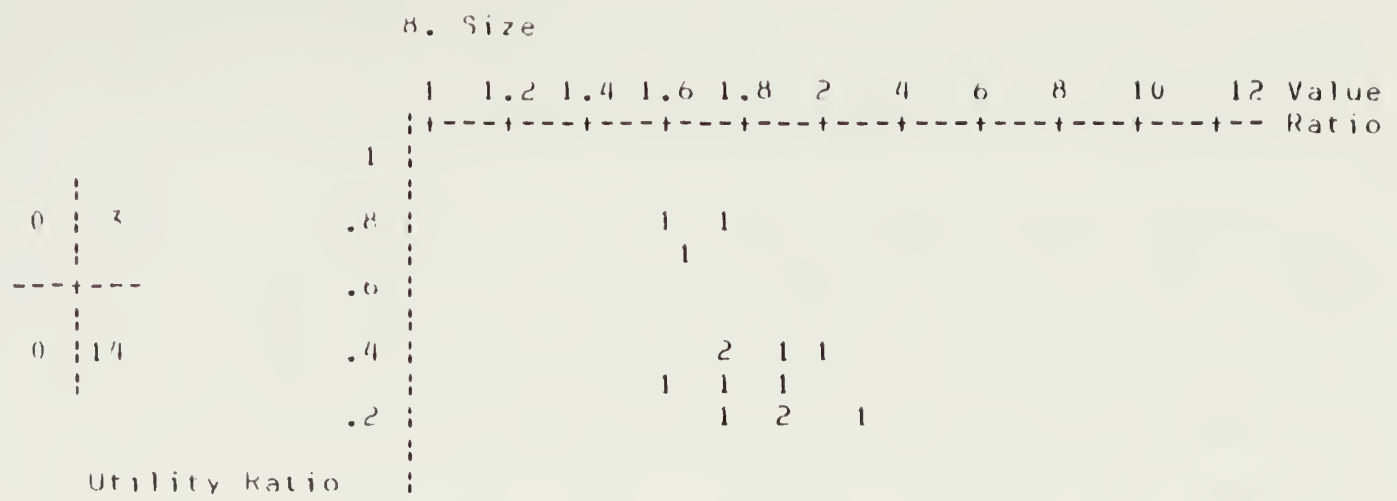
ATTRIBUTE LIST

	changes		changes
1 Fuel Consumtion (miles/gal.)	3	3 Maintainability	4
2 Speed	0	4 Comfort	8

The above list tells you that 8 people have entered data for the attribute, COMFORT, since you last entered data for that attribute. Since many changes have taken place you may want to see the display again and reassess your own input considering the new feedback. The choice is yours.

Enter your request (n,p or i) :

The example below shows a typical display of the type of information you will see when you request feedback for a given attribute :



Do you want to input data y/n (h for help) ?

Enter your request (n,p or j) :

EXPLANATION OF THE DISPLAY :

The display on the left gives the number of data points found in each quadrant. If several points are in a quadrant different from the one you selected, you might want to see data in other quadrants or change your own quadrant.

The points on the graph at the right show the value and utility ratios data. An integer k indicates that k subjects provided the same values.

The entry 2 on the graph at the intersection of the value ratio of 1.75 and the utility ratio of .4 means that two members of the assessment team input the pair (1.75,0.4),(or values which round to that pair).
That input means that the two felt that the subject system was 1.75 times larger than the baseline system, and the utility for the SIZE of the subject system is only 0.4 of the utility for the SIZE of the baseline system.

The prompt below the display asks if you want to update your input, y for yes, n for no, h for help.
You may also request any of the options described in HELP.

Enter your request (n,p or j) :

Should you type h you will see :

INSTRUCTIONS
============

During a session when you are asked to input data,
you may answer (watch for capital letters) :

y - YES, and be prompted for data entry
n - NO, and be prompted to choose another attribute
a - To get the ATTRIBUTE LIST
q - To change the Quadrant
h - HELP, to get this list
I - To restart at the Introduction
F - END of session

Enter any of the above or c to continue

Should you type any other character you will get this list to remind
of the valid responses !

Enter your request (n,p or ;) :

More detailed explanations of the commands (a,q,I,E,c) follow :

a - you will get back the attribute list and be able to select another attribute to work with, without entering any data on this one.

q - will let you look at the data in other quadrants at your choice. You will be prompted to input the quadrant number, the data will be displayed immidiatly in the new format as the response. You can use q again and pick the old or any other quadrant.

I - will get you into this introductory phase again so you will be able to flip pages and clear anything you need.

E - will terminate the session and will stop the execution of the program

c - will let you chose a new attribute without using the attribute list.

Enter your request (n,p or j) :

This completes the introductory phase. You may now type j and begin with the first iteration of data input or you may go back through parts of the introduction (by typing p or n) until you fill that you understand the instructions. Remember that you can access the introduction at any time through the help.

Enter your request (n,p or j) :

After the user has toured the introductory phase at his own and he feels that he understands the instructions he should type "j" to begin the actual utility assessment and feedback process. The followimg pages illustrate sample display information and data entry for the first iteration of the process.

MULTIATTRIBUTE UTILITY DETERMINATION
====================================

For the first iteration you have to fill in the quadrant proficiency information, and the initial value and utility ratios for each attribute.

Fuel Consumption (miles/gal.) Proficiency =
                               Quadrant   =

After the requested information is input the proper quadrant will be displayed.

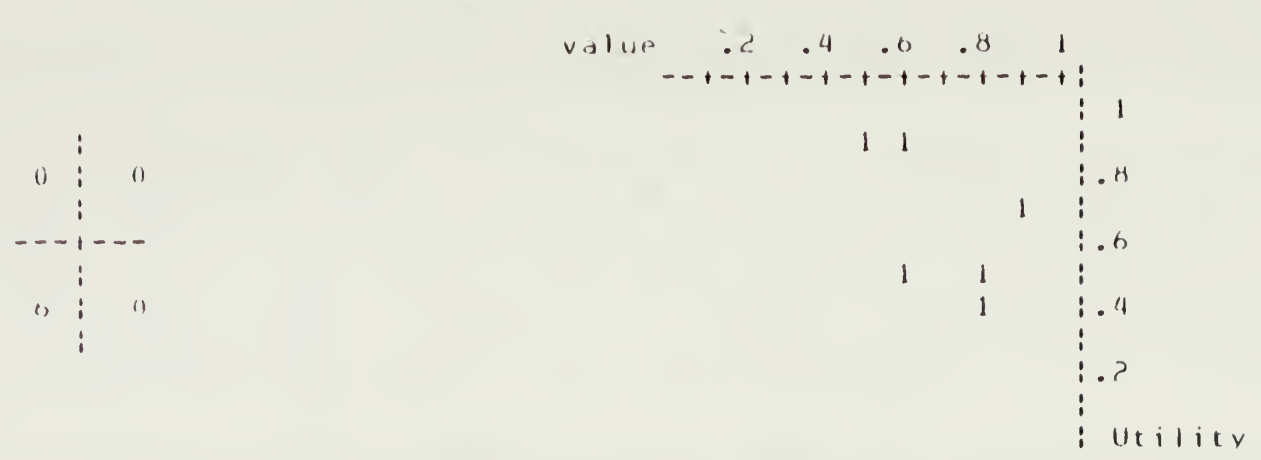
```
t2     Page 1     Wed Sep 12 21:38:25 1979

                    1 Fuel Consumption (miles/gal.)

          Utility
                        10
                         8
                         6
                         4
                         2
    0     |   0
                       1.8
    ------|---
                       1.6
    0     |   0
                       1.4

                       1.2

                         1
                        +---+---+---+---+---+---+---+---+---+---+---
                        1   1.2 1.4 1.6 1.8  2   4   6   8  10  12  Value

Enter value ratio :
Enter utility ratio :
```

When a user has entered the quadrant, the self-proficiency rating, and the utility and value ratios for each attribute the first iteration is complete. The program will then automatically transfer the user into the iterative feedback phase which allow the user to see the input provided by all of the the users for each attribute and to make changes to his own entries if he desires. The user is completly on his own to iterate as many times as he wishes.
The type of feedback he is provided is shown on the following two pages :

ATTRIBUTE LIST
==============

	changes		changes
1 Fuel Consumption (miles/gal.)	0	3 Maintainability	0
2 Speed	0	4 Comfort	0

Enter the # of the att. you want. 0 to get the attribute list



Do you want to input data y/n (h for Help) ?

When a user is satisfied with the all of his input, he should enter F to terminate the session.

The first program to be used by the manager-monitor is "atlst". Through this program an attribute list is created or updated. The list is kept in the file "atlist" from which it can be read by a program.

The program "clear is used to clear and initialize the data base files "atb" and "bnk". When the users are ready to start, the COMMAND "tbox < Y&" will begin execution of the administrator program. In the command, "Y" is a file that contains the capital letter "Y", as an answer to the "tbox" question whether this is a new run or a continuation. For a continuation "Y" should not be given. If "Y" is not given the saved "tbox" image of the previous process is copied and execution continues from the previous end point. Once "tbox" is running in the background the users can log in and implement the group decision making procedure.

While the users are going through their tasks the manager may run the analysis and monitoring program "an". This program is used to check the inputs to the data base as it is created. The following sample runs demonstrate the use of "an".

At the end of a session when all users have logged out, the manager should kill the background process "tbox" by executing the "boxstop" program. The data base is converted into a format compatible with an IBM computer for the purpose of analysis by running the program "ll". This program converts the data base into "long integers" and changes the format.

The command file "tape" contains the proper commands to load the data base onto a 9-track tape.

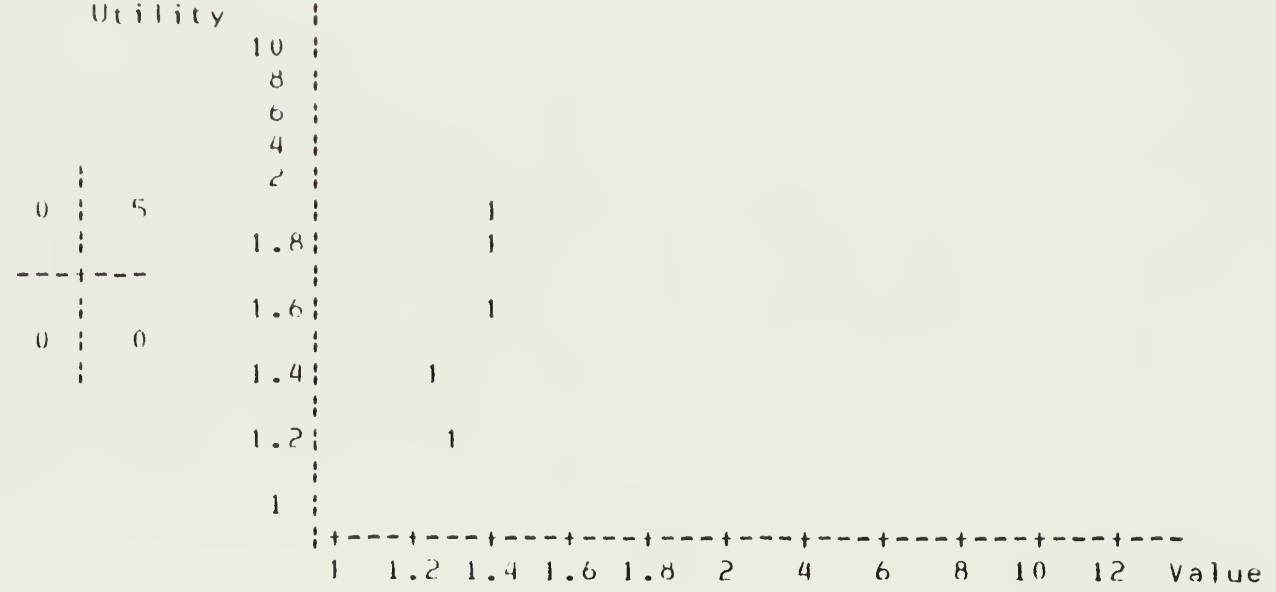
The FORTRAN program "tapdsk" and the CP/CMS exec routine that runs it are used to read the data bank files from a tape into CMS files that can then be handled by APL software.

```
t1     Page 1     Wed Sep 12 21:45:40 1979

Enter user # , 99 for all , 66 for time cuts , NEGATIVE to quit.    99

Enter attribute # , 99 for all      99

un=   1   atn=   1   prof= 4   quad= 1   valr= 1.40   utyr= 1.80   t0t1= 4660 -16083
un=   1   atn=   2   prof= 5   quad= 3   valr= 0.90   utyr= 0.70   t0t1= 4660 -16073
un=   1   atn=   3   prof= 3   quad= 2   valr= 0.70   utyr= 0.50   t0t1= 4660 -16062
un=   1   atn=   4   prof= 5   quad= 3   valr= 0.50   utyr= 0.20   t0t1= 4660 -16052
un=   2   atn=   1   prof= 4   quad= 1   valr= 1.40   utyr= 1.60   t0t1= 4660 -15976
un=   2   atn=   2   prof= 5   quad= 3   valr= 0.80   utyr= 0.50   t0t1= 4660 -15955
un=   2   atn=   3   prof= 4   quad= 3   valr= 0.70   utyr= 0.50   t0t1= 4660 -15930
un=   2   atn=   4   prof= 5   quad= 3   valr= 0.40   utyr= 0.40   t0t1= 4660 -15907
un=   3   atn=   1   prof= 4   quad= 1   valr= 1.30   utyr= 1.25   t0t1= 4660 -15793
un=   3   atn=   2   prof= 4   quad= 3   valr= 0.60   utyr= 0.50   t0t1= 4660 -15782
un=   3   atn=   3   prof= 5   quad= 2   valr= 0.80   utyr= 1.20   t0t1= 4660 -15771
un=   3   atn=   4   prof= 3   quad= 1   valr= 0.00   utyr= 0.00   t0t1= 4660 -15759
un=   4   atn=   1   prof= 4   quad= 1   valr= 1.40   utyr= 1.90   t0t1= 4660 -15518
un=   4   atn=   2   prof= 5   quad= 3   valr= 0.80   utyr= 0.40   t0t1= 4660 -15482
un=   4   atn=   3   prof= 4   quad= 3   valr= 0.50   utyr= 0.50   t0t1= 4660 -15443
un=   4   atn=   4   prof= 5   quad= 1   valr= 0.00   utyr= 0.00   t0t1= 4660 -15395
un=   5   atn=   1   prof= 4   quad= 1   valr= 1.25   utyr= 1.40   t0t1= 4660 -15542
un=   5   atn=   2   prof= 4   quad= 3   valr= 0.50   utyr= 0.90   t0t1= 4660 -15502
un=   5   atn=   3   prof= 4   quad= 2   valr= 0.50   utyr= 1.10   t0t1= 4660 -15450
un=   5   atn=   4   prof= 5   quad= 3   valr= 0.80   utyr= 0.40   t0t1= 4660 -15415
un=   6   atn=   1   prof= 0   quad= 1   valr= 1.50   utyr= 1.40   t0t1= 4660 -15183
un=   6   atn=   2   prof= 0   quad= 3   valr= 0.60   utyr= 0.90   t0t1= 4660 -15170
```

```
t2    Page 1    Wed Sep 12 21:46:44 1979

Enter user # , 99 for all , 66 for time cuts , NEGATIVE to quit.   99

Enter attribute # , 99 for all     1

un=  1  atn=  1  prof= 4  quad= 1  valr= 1.40  utyr= 1.80  t0t1= 4660 -16083
un=  2  atn=  1  prof= 4  quad= 1  valr= 1.40  utyr= 1.60  t0t1= 4660 -15976
un=  3  atn=  1  prof= 4  quad= 1  valr= 1.30  utyr= 1.25  t0t1= 4660 -15793
un=  4  atn=  1  prof= 4  quad= 1  valr= 1.40  utyr= 1.90  t0t1= 4660 -15518
un=  5  atn=  1  prof= 4  quad= 1  valr= 1.25  utyr= 1.40  t0t1= 4660 -15542
un=  6  atn=  1  prof= 0  quad= 1  valr= 1.50  utyr= 1.40  t0t1= 4660 -15183
```

```
t3    Page 1    Wed Sep 12 21:50:10 1979

Enter user # , 99 for all , 66 for time cuts , NEGATIVE to quit.    66

Enter attribute you want the cut for :    1

Enter the quadrant for display :    1

Enter the proficiency level :    1

Time now is -13638, enter the time cut :  -13638
```

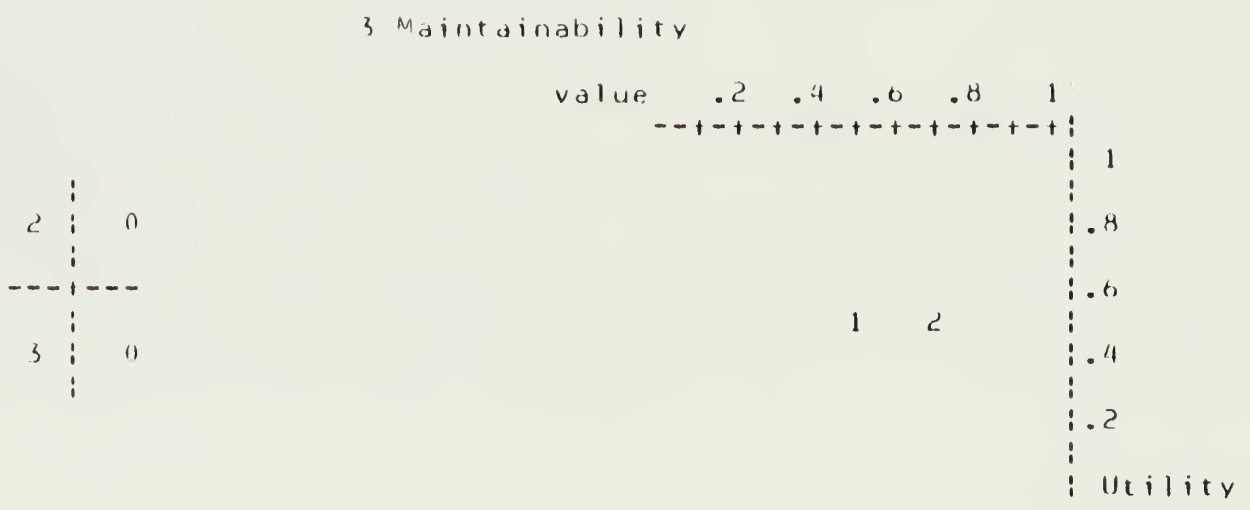
1 Fuel Consumption (miles/gal.)



```
t4    Page 1    Wed Sep 12 21:56:14 1979

Enter user # , 99 for all , 66 for time cuts , NEGATIVE to quit.     66

Enter attribute you want the cut for :    1

Enter the quadrant for display :     1

Enter the the proficiency level :    1

Time now is -13274, enter the time cut :   -15293

                    1 Fuel Consumption (miles/gal.)

        Utility     |
                 10 |
                  8 |
                  6 |
                  4 |
                  2 |
    0   |   3       |
        |           |
        |        1.8|          1
    ----+---       |
        |           |
        |        1.6|          1
    0   |   0       |
        |           |
        |        1.4|
                    |
                 1.2|       1
                    |
                  1 |
                    +---+---+---+---+---+---+---+---+---+---+---
                    1  1.2 1.4 1.6 1.8  2   4   6   8  10  12  Value
```

```
t5    Page 1    Wed Sep 12 21:53:55 1979

Enter user # , 99 for all , 66 for time cuts , NEGATIVE to quit.     99

Enter attribute # , 99 for all     3

un=  1  atn=  3  prof= 3  quad= 2  valr= 0.70  utyr= 0.50  t0t1= 4660 -16062
un=  2  atn=  3  prof= 4  quad= 3  valr= 0.70  utyr= 0.50  t0t1= 4660 -15930
un=  3  atn=  3  prof= 5  quad= 2  valr= 0.80  utyr= 1.20  t0t1= 4660 -15771
un=  4  atn=  3  prof= 4  quad= 3  valr= 0.50  utyr= 0.50  t0t1= 4660 -15443
un=  5  atn=  3  prof= 4  quad= 2  valr= 0.50  utyr= 1.10  t0t1= 4660 -15450
```

```
to    Page 1    Wed Sep 12 21:55:58 1979
Enter user # , 99 for all , 66 for time cuts , NEGATIVE to quit.    66
Enter attribute you want the cut for :     3
Enter the quadrant for display :       3
Enter the proficiency level :          1
Time now is -13128, enter the time cut :      -13128
```



t/ Page 1 Wed Sep 12 21:58:03 1979

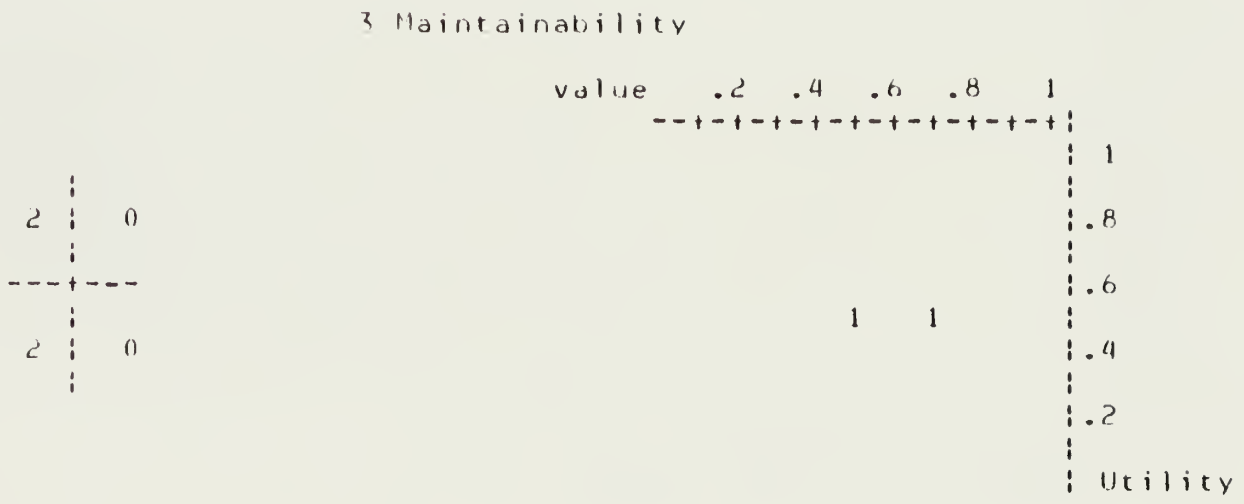
Enter user # , 99 for all , 66 for time cuts , NEGATIVE to quit. 66

Enter attribute you want the cut for : 3

Enter the quadrant for display : 3

Enter the proficiency level : 4

Time now is -13018, enter the time cut : -13218



t8 Page 1 Wed Sep 12 22:00:11 1979

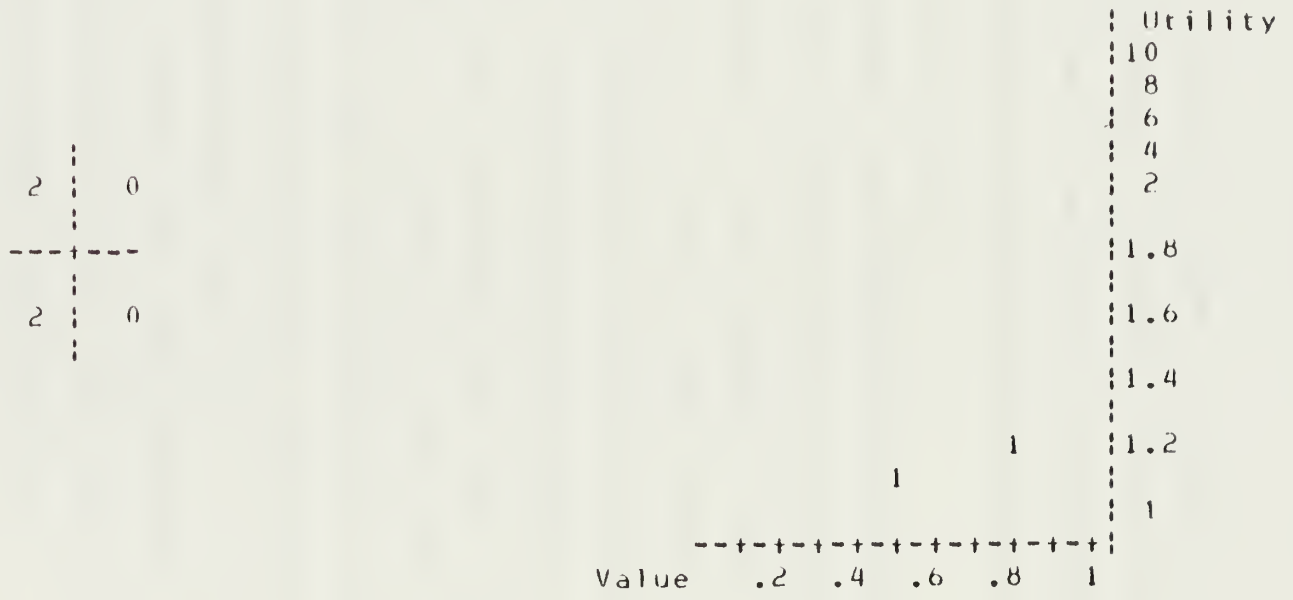
Enter user # , 99 for all , 66 for time cuts , NEGATIVE to quit. 66

Enter attribute you want the cut for : 3

Enter the quadrant for display : 2

Enter the proficiency level : 4

Time now is -12981, enter the time cut : -13018



Enter user # , 99 for all , 66 for time cuts , NEGATIVE to quit. -8

VI. LIMITATIONS AND EXTENSIONS

The current implementation of the group decision making procedure on the PDP 11/50 reserves half the available data core storage for the user's input data records. Some extra core can still be allocated for that purpose but not a significant amount.

The current limit on the total number of records is 2K. Any number of users, attributes, and inputs per user per attribute whose product does not exceed 2K is acceptable. In order to be able to handle a significantly greater amount of data one could page the data bank.

In the current application the graphical display used is a "dumb" CRT terminal. The procedure would be enhanced by using a split-screen terminal or a full graphic display terminal. Only minor changes to the main user program "tt.c" (in the graphical display subroutines) would be required to adapt the application to a more sophisticated terminal.

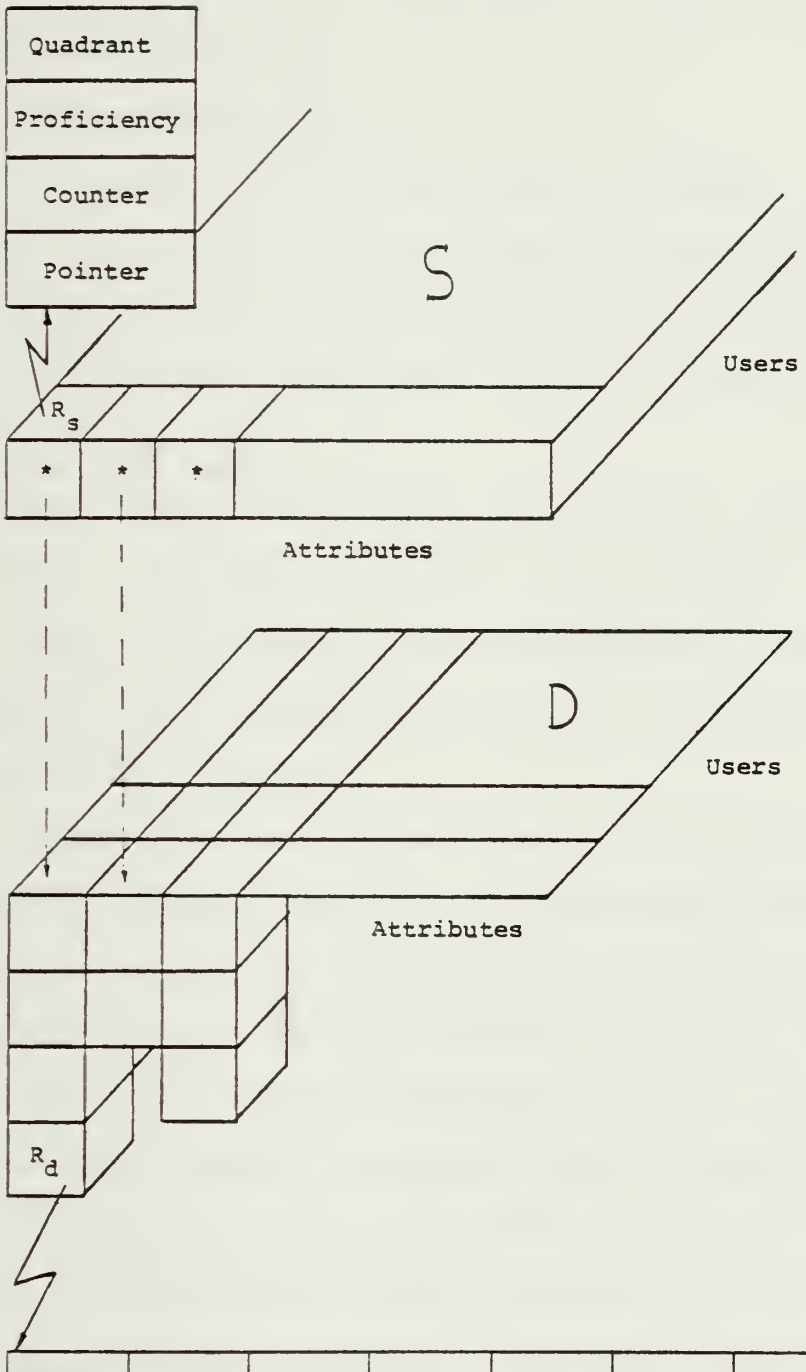
The procedure can be used to evaluate more than one alternative and any number of scenarios by saving the data bank files under different names and reinitializing the procedure for each new alternative-scenario combination.

The limited data base management and analysis system described in this thesis for the group decision making procedure can easily be adapted for several other applications. The basic features that the process provides are:

1. Multiple users simultaneously assessing a common dynamic data base.
2. Immediate selected feedback with anonymity provided to the users.
3. Protection of the data base so that one user's inputs are never allowed to overwrite another's inputs.
4. Complete visibility of the actions that take place to a monitor or umpire who could communicate with any user.
5. Users allowed to proceed at their own pace.
6. Complete internal self-documentation with system prompts requesting all necessary data.
7. A capability of reconstructing the entire process over time of each user.

These features are useful for various types of information gathering or decision making tasks. One especially important application is to the area of interactive war gaming.

CONCEPTUAL DATA STRUCTURE



quadrant when entered	pointer to previous record	user number	attribute number	value ratio	utility ratio	time 1	time 2

FIGURE 5

APPENDIX A

SOFTWARE DESIGN AND IMPLEMENTATION

A. THE DATA STRUCTURE

1. Conceptual View

Two major types of data are invovled: static and dynamic. The static data includes a fixed amount of information for each user-attribute pair. The dynamic data consists of the record built from each value and attribute ratio entered by the users.

Figure 5 depicts the static data body S, the dynamic data body D and their respective data records R_s and R_d. There are (# of users) x (# of attributes) records R_s. The contents of the information in R_s may change but the locations of the fields are fixed.

In the body D the top horizontal plane contains the last information entered about an attribute-user pair in records of the form R_d. When a user updates the values for an attribute, the new data is placed on the top of the stack that grows down from the horizontal plane. At any given time the user program has access only to the data records that forms the top horizontal plane (the top of the stacks).

The program monitor needs access to the entire data base accumulated during a session. For every user-attribute pair he needs to be able to see the self-rated proficiency, and the quadrant for display, then all the value and utility

ratios that were entered and the time of entry. Later, after the data are gathered, the analyst will need to reconstruct the data input history in order to be able to investigate the input pattern of all the users and thus be able to detect any biases if they exist.

2. The Implementation

The "C" language was selected because of its pointer handling facilities, its ability to easily define and operate on complex data structures, and its simple handling of character information.

A data record R_d consists of the following fields:

- user number
- attribute number

} the indexing pair

- attribute value ratio
- utility ratio

} the basic user's input

- time 1
- time 2

} system clock time at data entry

- pointer -- a link to the previous record entered by the user for an attribute

For each indexing pair (user number; attribute number) a record R_s will contain the fields:

- pointer -- a link to the corresponding stack-top in D
- counter -- the number of changes made to this attribute since this user last updated it

- proficiency -- the self-rated proficiency of the user with respect to this attribute
- quadrant -- the quadrant in which the next display will be presented.

The structure:

```
struct attr{
     int user[NUSERS],count[NUSERS],prof[NUSERS],quadr[NUSERS];
    }
```

contains all the R_s records for a certain attribute. S consists of R such structures where k is the number of attributes.

The structure:

```
struct data{
     int dm,  pd; usn, atrn, usl, vsl, time[2];
    }
```

contains a data record R_d. An array of 2K such records is defined and contains the data bank; it is large enough to allow each of 20 users to input 4 different pairs for each of 25 attributes. This should be sufficiently large to accomodate most applications. Since individual users will differ in updating patterns, no attempt is made to allocate

a fixed number of blocks to each user. Instead, records from the bank are allocated sequentially.

The structures that contain S are stored and maintained in the file "atb" and the ones that form D in the file "bnk". Both will be referred to as the data bank.

Searching and updating the data bank on the files is a very time consuming I/O operation, hence, periodically the files are read into the appropriate structures in core and the searching is performed in core.

Note: Since the stacks in D are maintained as singly linked lists, no updates are needed in the previous records when a record is added on the top of a stack. The pointer field in the new record will point to the old top, and the only update is done in the pointer field of the proper record in the structure "attr" that now will point to the new top.

B. DATA INTEGRITY

Since UNIX is a time sharing system all communications and all data shared by two or more users must take place through use of the file system. At any time only one user has access to the CPU. Care must be taken that the data in the files that are shared by multiple users is protected so that the input from one user cannot write over another's data.

The data which we collect are stored in two files. The static data base (contained in the structures "att") are stored in the file called "atb". This file has a fixed size and each field has a fixed predetermined location. Therefore two different users will always write into different locations.

Protection of this data file is therefore no problem. A simple algorithm in the user's program takes care to write all information in the "atb" file in the proper locations.

The dynamic portion of the data base is contained in the file "bnk". Space is not preallocated in this file, nor is specific information stored in predetermined locations. New information is appended to the data base without erasing any old data. The software must keep track of the size of the file and the location of the next record to be added to the data base. At any instant two or more users may simultaneously decide to add new data to the file and the same location may be given to multiple users. Therefore the input from one user could write over the input of another user, and the former value would be lost forever. Therefore a protection mechanism was developed to make sure that the problem never occurs.

C. THE TICKET BOX SOLUTION

A sequencing approach described in Ref. [11] was adapted to the problem at hand to protect the data base from one user writing over the data of another user. The approach is

analogous to the method used increasingly by many businesses to accommodate a multi-server, single queueing process. A customer arriving at a store desiring service is assigned a numbered ticket. In front of the waiting customers is a display showing the number of the customer presently receiving service. When a service is completed the display counter is incremented by one and the waiting customer holding the matching ticket is serviced.

The same principle is used to provide the data base protection in our problem. It is handled with an administrative program called "tbox". It runs concurrently with the users' programs receiving write requests from the users, issuing sequential tickets, incrementing the counter when a write is completed and checking for matches between tickets and the counter. Its operation is described by the flowchart in Figure 6.

The files called "REQUEST" and "TICKET" contain a fixed number of fields equal to the number of users. The file "COUNT" contains a single integer.

Each user who wants to write new information into the data base first requests permission to write through the file "REQUEST". A check is then made of the "TICKET" file to determine if the administrative program "tbox" has issued a ticket to the user. This check is repeated until a ticket has been issued. Once a ticket is issued a companion of the ticket number is made with the "COUNT" file to determine if

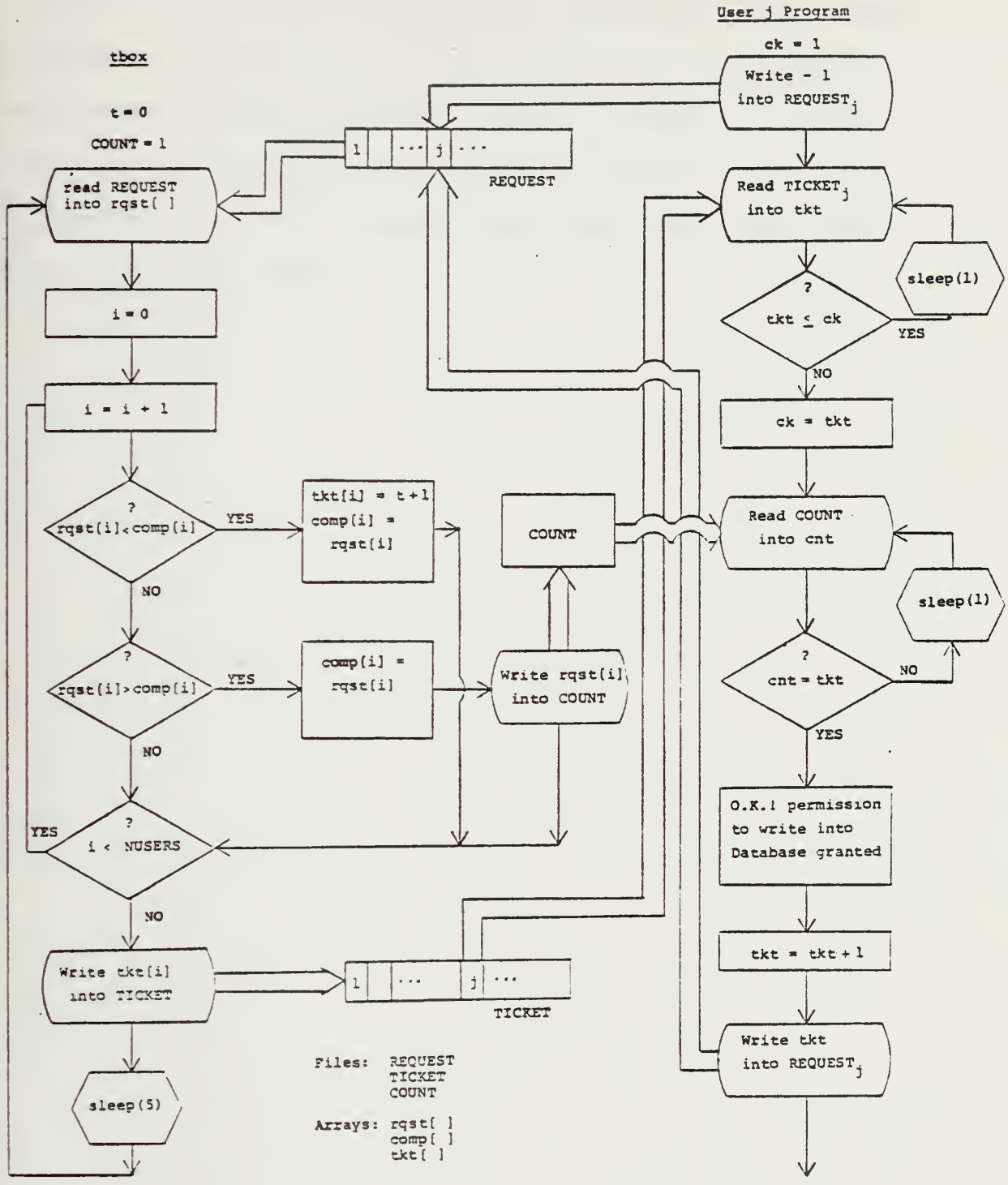

User j Program
tbox
t = 0
COUNT = 1
read REQUEST into rqst[ ]
i = 0
i = i + 1
rqst[i]<comp[i]
YES
tkt[i] = t+1
comp[i] = rqst[i]
NO
rqst[i]>comp[i]
YES
comp[i] = rqst[i]
NO
i < NUSERS
YES
NO
Write tkt[i] into TICKET
sleep(5)
1 ... j ...
REQUEST
TICKET
COUNT
Write rqst[i] into COUNT
ck = 1
Write - 1 into REQUEST_j
Read TICKET_j into tkt
sleep(1)
tkt ≤ ck
YES
NO
ck = tkt
Read COUNT into cnt
sleep(1)
cnt = tkt
NO
YES
O.K.! permission to write into Database granted
tkt = tkt + 1
Write tkt into REQUEST_j
Files: REQUEST
TICKET
COUNT
Arrays: rqst[ ]
comp[ ]
tkt[ ]

FIGURE 6

the user has write permission. When the write operation is completed the user's program sends a message to the administrator to increment the counter so new write permission can be granted. The programs are sent to "sleep ( )" when a check fails since there is no sense in wasting the time in checking; no other program that can change the status runs at the same time.

```
 1
 2 //          GROUP DECISION MAKING WITH FEEDBACK
 3 //          ==================================
 4 //
 5 //          The user's program.
 6 //          -------------------
 7 //
 8 #define NUSERS 16   // maximum number of users + 1
 9 # define NATTR  28  // maximum number of attributes + 1
10 #define BNKSIZE 16  // size of a data record in bytes
11
12 //  structure data defines the basic data record. the fields are:
13 //
14 //  dm   - display quadrant at data entry time
15 //  pd   - pointer to the privious data record by the same user
16 //       on the same attribute
17 //  usn  - the user unique ID number
18 //  atrn - the attribute number
19 //  vsl  - the value ratio entered
20 //  usl  - the utility ratio entered
21 //  time[2] - system time at data entry
22
23 struct data {
24     int dm, pd, usn, atrn, vsl, usl, time[2];
25 };
26
27 //  the structure attr holds the constant size information about
28 //  the users and the attributes.
29 //
30 //  usr   - pointer to the last data record entered by a given user
31 //          about a given attribute
32 //  count - tally of the number of changes to a given attribute
33 //          since last updated by a given user
34 //  prof  - the self-rated proficiency for each user-attribute pair
```

```
35 //  quadr - the quadrant the user choses to see the data in
36
37 struct attr {
38      int usr[NUSERS], count[NUSERS], prof[NUSERS], quadr[NUSERS];
39 };
40
41 struct data bank[2048];
42 struct attr att[NATTR];
43 struct data *B;
44
45 char d[16][45];
46 int q[NATTR][4];
47 int geti();
48 double getf();
49 char getchar();
50
51 char *HH[] { "\n\n\n\n\n\n\n\n\n",
52      "              INSTRUCTIONS                ",
53      "              ============                ",
54      "  ",
55      " During a session when you are asked to input data,",
56      " you may answer (watch for capital letters) : ",
57      "  ",
58      "  y - YES, and be prompted for data entry ",
59      "  n - NO,  and be prompted to choose another attribute ",
60      "  a - To get the ATTRIBUTE LIST ",
61      "  q - To change the Quadrant    ",
62      "  h - HELP, to get this list  ",
63      "  I - To restart at the Introduction ",
64      "  E - END of session ",
65      "   ",
66      "Enter any of the above or c to continue \n\n\n\n ",
67      0 };
68
```

```
 69 char *all "/usr/tamir/tpq0/bnk";     // links to the files common to all users
 70 char *al2 "/usr/tamir/tpq0/atb";
 71 char *alt "/usr/tamir/tpq0/atlist";
 72 char *alc "/usr/tamir/tpq0/count" ;
 73 char *alr "/usr/tamir/tpq0/request";
 74 char *alk "/usr/tamir/tpq0/ticket" ;
 75 char *acc "count" ;
 76 char *acr "request" ;
 77 char *ack "ticket" ;
 78 char *ac1 "bnk";
 79 char *ac2 "atb";
 80 char *act "atlist";
 81 int M -1;
 82 int fda, fdb, fdc, fdr, fdk;
 83 int un, atn, atsize, ch, natt, fsize ;
 84 char c;
 85 char S[40][32]; // will hold the attribute list created
 86                 // by the program 'atlst'.
 87
 88 //  The main program handles the flow of the process. It performs the
 89 //  the linking, opening and unlinking of the various files.
 90 //  A unique user number is associated with each user and the basic
 91 //  loop of the program is entered.
 92
 93 main() {
 94     int i, j, fat;
 95     atsize = 8*NATTR*NUSERS;
 96     B = &bank;  // Sets the base address of the databank.
 97     unlink(c1);
 98     unlink(c2);
 99     unlink(ct);
100     unlink(ck);
101     unlink(cr);
102     unlink(cc);
```

```
103     link(l1, c1);
104     link(l2, c2);
105     link(lt, ct);
106     link(lc,cc) ;
107     link(lr,cr) ;
108     link(lk,ck) ;
109     fdc = open(cc,2) ;
110     fdr = open(cr,2) ;
111     fdk = open(ck,2) ;
112     fda = open(c2, 2);
113     fdb = open(c1, 2);
114     fdt = open(ct, 2);
115     seek(fda, 0, 0);
116     read(fda, att, atsize);
117     seek(fdt, 0, 0);
118     read(fdt, &natt, 2);     // Read the number of attributes and
119     read(fdt, S, 1280);      // the attribute list itself (as prepared by 'atlst').
120
121
122     i = getuid()&0377;  // Get a unique user number for each user from
123     un = i-141; // the password file. The 141 needs to be changed if
124                 // used in another system.
125
126     seek(fdk,2*un,0) ;  // Sets the check value for ticket verification
127     read(fdk,&ch,2)  ;  // in case it is a follow on session.
128
129     intro();
130     prep();
131     if(c != 'j')
132         menu();
133     do {
134         do {
135         printf("\nEnter the # of the att. you want. Enter ");
136         printf("0 to get the attribute list. \n") ;
```

```
            atn = geti();
            if(atn == 0) { menu() ; atn = -1 ; }
            }
            while(!(atn >= 0 && atn <= (natt-1)));
            attrib();
        }
        while(c != 'F');


        unlink(c1);
        unlink(c2);
        unlink(ct);
        unlink(ck);
        unlink(cr);
        unlink(cc);

    //  Unlinking should be done externaly if the promgram terminates
    //  abnormaly .
    }

    //  The prep routine performs the first iteration and asks the user
    //  for two basic pieces of information :
    //      1. The user's self-rated proficiency.
    //      2. The quadrant the user chose to see the display initialy.
    //  Proficiency information may not be changed, however the quadrants
    //  may be updated later at any time.
    //  After the entry of the above the quadrant is displayed and the user
    //  is asked to input the value and utility ratios.
    //  This sequence is repeated for all attributes.

    prep() {
        int i, j, ip, iq, ul,vl ,k,l ;
        float fvl,ful ;
        printf("\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n") ;
```

```
171     printf("\t\tMULTIATTRIBUTE UTILITY DETERMINATION \n");
172     printf("\t\t================================== \n\n");
173     printf("\tFor the first iteration you have to fill in the quadrant\n");
174     printf("   proficiency information, and the initial value and \n");
175     printf("   utility ratios for each attribute. \n\n" ) ;
176     for(i = 1; i < natt; i++) {
177         printf("\n%s  Proficiency = ", S[i]);
178         while(j <= 0 || j > 5) {
179             j = geti();
180             if(j == 99)     // One may skip this phase by using 99
181                 goto escape;// for proficiency.
182         }
183         att[i].proffun] = j;
184         j = 0;
185         printf("\n\t\t\t\t\t\t\t\t Quadrant    = ");
186         while(j <= 0 || j > 4)
187             j = geti();
188
189         att[i].quadr[un] = j;
190         atn = i ;
191         for(k = 0; k < 16; k++) {   // Initialize the display array
192             for(l = 0; l < 44; l++)
193                 d[k][l] = ' ';
194             d[k][44] = '\0';
195         }
196         for(k = 0; k < NATTR; k++)  // initialize the quadrant indicator array.
197             for(l = 0; l < 4; l++)
198                 q[k][l] = 0;
199         switch(j) {     // Display the proper quadrant.
200             case 1 : graf1() ; break ;
201             case 2 : graf2() ; break ;
202             case 3 : graf3() ; break ;
203             case 4 : graf4() ;
204         }
```

```
205             fvl = ful = -1 ;
206                 while(fvl < 0) {
207                     printf("\nEnter value ratio :   ");
208                     fvl = getf();
209                 }
210                 while(ful < 0) {
211                     printf("Enter utility ratio : ");
212                     ful = getf();
213                 }
214                 vl = fvl*1000;
215                 ul = ful*1000;
216                 indata(vl, ul);
217             j = 0;
218         }
219 escape:
220         for(j = 0; j < i; j++) {
221             jp = NUSERS*(8*j+4)+2*un;    // Compute addresses in the static files
222             jq = NUSERS*(8*j+6)+2*un;
223             seek(fda, jp, 0);
224             write(fda, &attij.prof[un], 2); // write in the preliminary data
225             seek(fda, jq, 0);
226             write(fda, &attij.quadr[un], 2);
227         }
228 }
229
230 //  The menu routine display to the user the list of attributes and a
231 //  number by which the attribute will be referenced.
232 // Under the column 'changes' the count of the number of times other users
233 //  have updated utility information since he last entered data for it.
234
235 menu() {
236         int j;
237         seek(fda, 0, 0);
238         read(fda, att, atsize);
```

```
239     printf("\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n") ;
240     printf("\n\n\n\n\n\t\t\t\t\t\t\tATTRIBUTE LIST \n");
241     printf("\t\t\t\t\t\t\t============== \n\n");
242     printf("\t\t\t\t\t\t\tchanges\t\t\t\t\t\t\tchanges\n");
243     for(j = 1; j <= natt/2; j++)
244         printf(" %2d %s %2d     %2d %s %2d \n", j, S[j], att[j].count[un],
245                 (natt/2 + j), S[natt/2+j], att[natt/2+j].count[un]) ;
246
247     return;
248 }
249
250 //  The attrib routine will disolay to the user a requested attribue
251 //  in the proper quadrant, will prompt for value and utility ratio
252 //  entry and will make available the help function.
253
254 attrib() {
255     struct data *p ;
256     float fv1, fu1;
257     int v1, u1;
258     int i, jo;
259     i = 0;
260     fv1 = fu1 = -1.0;
261     outdata();
262
263   msg :
264     printf("\nDo you want to input data y/n ( h for Help ) ? \n ") ;
265     c = getchar(); getchar() ;
266     while(c != 'c') {
267         switch(c) {
268         case 'y':
269             while(fv1 < 0) {
270                 printf("Enter value ratio :   ");
271                 fv1 = getf();
272             }
```

```
273             while(ful < 0) {
274                 printf("Enter utility ratio : ");
275                 ful = getf();
276             }
277             vl = fvl*1000;  // Integerize the data.
278             ul = ful*1000;
279             indata(vl, ul);
280             return;
281         case 'n':
282             return;
283         case 'a':
284             menu();      // display attribute list.
285             return;
286         case 'q':   // Lets the user change the display quadrant.
287             printf("Enter Quadrant :  ");
288             while(i <= 0 || i > 4)
289                 i = geti();
290             att[atn].quadr[un] = i;
291             jq = NUSERS*(atn*8+6)+2*un;
292             seek(fda, jq, 0);
293             write(fda, &att[atn].quadr[un], 2);
294             attrib();
295             return;
296         case 'h':
297             while(H[i] != 0)
298                 printf("%s\n", H[i++]);
299             c = getchar();
300             break;
301         case 'M' :                  // Prints the user's own data point values
302             p = B + att[atn].usr[un] ; // Not !! in the user's documentation.
303             fvl = (p -> vsl) / 1000.0 ; // Refloat the data.
304             ful = (p -> usl) / 1000.0 ;
305             printf("My point is: Value ratio = %2.2f  Utility ratio = %2.2f",
306                     fvl,ful) ;
```

```
307                 fvl = ful = -1.0 ;
308                 goto msg ;
309             case 'I':
310                 intro();
311                 menu();
312                 return;
313             case 'f':
314                 return;
315             default:
316                 c = 'h';
317             }
318         }
319         return;
320 }
321
322 //  The outdata routine generates a CRT display of the most recent
323 //  data entered by all the users in any selected attribute. The data
324 //  is displayed to each user in the quadrant which he selects.
325 //  A fresh copy of the data is read from the file system and the
326 //  search for the utility and value ratios is done in core.
327
328
329 outdata() {
330     struct data *p;
331     float vl, ul;
332     int i, k, l, dv, du;
333
334     vl = ul = 1.0;
335
336     for(k = 0; k < 16; k++) {    // Initialize the data display array.
337         for(l = 0; l < 44; l++)
338             d[k][l] = ' ';
339         d[k][44] = '\0';
340     }
```

```
341     for(k = 0; k < NATTK; k++)  // initialize the quadrant indicator array
342         for(l = 0; l < 4; l++)
343             qlk[l] = 0;
344
345     seek(fda, 0, 0);
346     read(fda, att, atsize);  // Read in a fresh copy of the data.
347     seek(fdb, 0, 0);
348     read(fdb, bank, 32767);
349     i = att[atn].quadrfun;
350     for(l = 0; l < NUSERS; l++) {
351         k = att[atn].usr[l];
352         p = B+k;
353         if(k == M)
354             continue;
355         else
356             v1 = (p->vs1)/1000.0+0.001;  // Refloat the ratios.
357         u1 = (p->us1)/1000.0+0.001;
358         if(v1 >= 1 && u1 >= 1)  // Compute the proper quadrant the data is in
359             qlatn[0]++;         // and prepare the quadrant indicator array
360         if(v1 < 1 && u1 >= 1)
361             qlatn[1]++;
362         if(v1 < 1 && u1 < 1)
363             qlatn[2]++;
364         if(v1 >= 1 && u1 < 1)
365             qlatn[3]++;
366         switch(i) {     // call the proper data display preparation routine
367         case 1:
368             q1(v1, u1);
369             break;
370         case 2:
371             q2(v1, u1);
372             break;
373         case 3:
374             q3(v1, u1);
```

```
375                 break;
376             case 4:
377                 q4(v1, u1);
378             }
379         }
380         switch(i) {       // Call the proper graphical display routine
381         case 1:
382             graf1();
383             break;
384         case 2:
385             graf2();
386             break;
387         case 3:
388             graf3();
389             break;
390         case 4:
391             graf4();
392         }
393         return;
394 }
395
396 //  The indata routine creates the new data records and writes them
397 //  into the bank on the file system.
398 //  Before writting the routine will look for permission to write into
399 //  the data base. As soon as permission is given, the information
400 // is written.
401
402 indata(v1, u1)
403         int v1, u1;
404 {
405         struct data *p;
406         int tvec[2];
407         int tp, j, jp,tkt,cnt ;
408         time(tvec);
```

```
409     cnt = tkt = 0 ;
410
411
412     seek(fdr,2 * un,0) ;
413     write(fdr,&M,2) ;    // Puts a write permission request from the "manager"
414     while(tkt <= ch) {
415         seek(fdk,2 * un,0) ;     // Gets a ticket and check if it is a ne one
416         read(fdk,&tkt,2) ;
417         sleep(1) ;  // Free the time slice
418 //printf(" tkt = %d  ch = %d \n",tkt,ch);
419       }
420     ch = tkt ;  // Set the check variable
421     while(cnt != tkt) {
422         seek(fdc,0,0) ; // Read the counter and check for your turn
423         read(fdc,&cnt,2) ; // ( when ticket = count )
424         sleep(1) ;  // free the time slice
425 //printf(" tkt = %d  cnt = %d \n",tkt,cnt);
426       }
427
428     // Write permission is now granted !
429
430     for(j = 0; j < NUSERS; j++)
431         attlatn].count[j]++;  // Increment changes counter for all other users
432     attlatn].count[un] = 0; // Reset user's own counter to 0
433
434     tp = attlatn].usr[un];
435     attlatn].usr[un] = cnt; // Get and compute the proper address
436     p = &tcnt;              // for the new record.
437     p->pd = tp; // Prepare the new record
438     p->vsl = v1;
439     p->usl = u1;
440     p->time[0] = tvec[0];
441     p->time[1] = tvec[1];
442     p->atrn = atn;
```

```
443     p->usn = un;
444     p->dm = att[atn].quadr[un];
445
446     seek(fdb, cnt * BNKSIZE, 0);
447     write(fdb, p, BNKSIZE); // Write the new record into the data base
448     jp = atn*NUSERS*8+2*un;
449     seek(fda, jp, 0);
450     write(fda, &att[atn].usr[un], 2);  // Set the pointer to the top of the
451     jp = NUSERS*(atn*8+2);             // stack.
452     seek(fda, jp, 0);
453     write(fda, &att[atn].count[0], NUSERS*2);
454     tkt = tkt + 1 ;
455     seek(fdr,2 * un,0) ;
456     write(fdr,&tkt,2) ; // Send the write coplete message to the administrator
457                         // through the request file.
458     return;
459 }
```

```
 1
 2 //  This separatly compiled subroutine will handle the display
 3 //  of the introduction phase.
 4
 5 //  The in.o object modul needs to be loaded with the program 'tt'.
 6
 7 char *P1[] {
 8 "   ",
 9 " ",
10 "            GROUP DECISION MAKING WITH FEEDBACK      ",
11 "            ==================================      ",
12 " ",
13 "   This program will help you as a member of the utility assessment group ",
14 "assign utilities to the attributes of the specified systems. \n ",
15 "We begin with an introduction which will provide instructions on",
16 "the use of the program. ",
17 "A page of information will be displayed. After you read and understand",
18 "the page you must respond by typing one of the following",
19 "characters and then a 'return' : ",
20 "   ",
21 "    n - To see the next page.  ",
22 "   ",
23 "    p - To see the previous page. ",
24 "   ",
25 "    j - To jump over the remaining introduction. ",
26 "    ",
27 "   ",
28 "   ",
29 "\n   ",
30 " Now enter your request   :   ",0 } ;
31
32 char *P2[] {
33 " "," ",
34 "FIRST ITERATION : \n ",
```

```
35 "   During the first iteration you will be asked to rate your own ",
36 "proficiency for evaluating the utility of each attribute by selecting an ",
37 "integer from 1 ( low ) to 5 ( high ). ",
38 "\n       1    2    3    4    5  ",
39 "       +----+----+----+----+   SELF-PROFICIENCY ",
40 "     low                 high  RATING  \n",
41 "You will not be able to change your self-proficiency rating for any ",
42 "attribute after you have entered it initially. Thus be careful in entering",
43 "the data.\n\n",
44 "   During the utility assessment process you will be shown a graph of ",
45 "utility ratio versus value ratio for each attribute. Because of screen",
46 "size and resolution limitation all four quadrants cannot be displayed.",
47 "Therefore you will be asked to select one of the four quadrants to be ",
48 "displayed for each attribute. ",
49 "\n",
50 "Enter request ( n,p or j ) :",0) ;
51
52 char *P51) {
53 "\n",
54 "                  utility",
55 "                  ratio      ¦",
56 "                           10+",
57 "                    2        ¦         1",
58 "                           2 +",
59 "                            ¦",
60 "                  --+----+----1----+----+--",
61 "                  .1    .5   ¦    2    10  value ratio",
62 "                           .5+",
63 "                       3     ¦         4",
64 "                           .1+",
65 "                            ¦",
66 "                         .05 +",
67 "\n for each attribute you will be shown the single quadrant which",
68 "you select. Unlike the self-proficiency rating which cannot be changed",
```

```
 69 "you may, at any time, change the quadrant which is displayed for an ",
 70 "attribute. ","\n\n",
 71 "\nEnter request ( n,p or j ) :",
 72 0};
 73
 74 char *P4[] {
 75 "\n\n\n\n\n",
 76 "   Comments about the Quadrants : \n ",
 77 "   We expect most of you to agree on the quadrant which contain the ",
 78 "utility data. If you think that large value ratios are preferred then",
 79 "you should select either quadrant 1 or quadrant 3. Quadrant 1 would be",
 80 "selected if you think that the subject system has more of the attribute",
 81 "than the baseline system ( i.e. value ratio >= 1 ); quadrant 3 if value",
 82 "ratio is < 1. If you think smaller value ratios are preferred then  ",
 83 "select either quadrant 2 or 4. ",
 84 "\n\n\n\n\n\n\n\n\n\n",
 85 "Enter request ( n,p or j )  : ",0 } ;
 86
 87 char *P5[] {
 88 "\n\n\n\n\n\n",
 89 "EXAMPLE : \n",
 90 "   The situation is transportation for a salesman who travels 40,000 ",
 91 "miles per year by automobile over mostly interstate highways. ",
 92 "The baseline system is the current midsize fleet vehicle owned by the",
 93 "parent company. The alternative system under consideration is a new",
 94 "vehicle produced by another auto manufacturer. ",
 95 "The attribute under consideration is COMFORT. Certainly, everything ",
 96 "else considered equal, more comfort is preferred. Therefore you should",
 97 "select quadrant 1 if you think that the alternative car is more comfor-",
 98 "table then the baseline. Select quadrant 3 if you think the baseline ",
 99 "is more comfortable. ",
100 "\n\n\n\n\n\n",
101 "Enter request ( n,p or j )  : ",0 } ;
102
```

```
103 char *P6[] {
104 "\n\n\n\n",
105 "   After you enter your self-proficiency rating and the quadrant number",
106 "for an attribute, the selected quadrant will be displayed, and you will",
107 "be asked to enter your assessment of both the value ratio and the utility",
108 "ratio for the attribute in the given scenario.",
109 "\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n",
110 "Enter request ( n,p or j ) : ",
111 0 } ;
112
113 char *P7[] {
114  "\n\n\n\n",
115 "FEEDBACK PHASE : \n",
116 "   After you have entered initial values for each of the attributes, you",
117 "will be allowed to display the data input by all of the team members for",
118 "any selected attribute. Anonymity will be maintained.",
119 "You will be able to see all of the entries, but you will not know who",
120 "entered a given data point.",
121 "You will be prompted to select an attribute with the following statement:",
122 "\n Enter the # of the att. you want. Enter 0 to get the att. list.",
123 "\nIf you type a number between 1 and n (n is the number of attributes) ",
124 "you will obtain a graphical display of the most recent data from all team",
125 "members for the selected attribute.",
126 "   If you enter 0 you will obtain the attribute list with each attribute",
127 "name, an identifying number, and a tally of the number of updates that",
128 "have transpired since you last entered data. See the example on the next page.",
129 "\n\nEnter request ( n,p or j ) : ",0 } ;
130
131 char *P8[] {
132 "\n\n\n",
133 "                              ATTRIBUTE LIST ",
134 "                              ============== \n\n",
135 "                                 changes                                changes",
136 " 1 Fuel Consumtion (miles/gal.)    3      3 Maintainability                 4",
```

```
137 " 2 Speed                                0       4 Comfort                                8",
138 "\n\n",
139 "   The above list tells you that 8 people have entered data for the ",
140 "attribute, COMFORT, since you last entered data for that attribute. ",
141 "Since many changes have taken place you may want to see the display",
142 "again and reassess your own input considering the new feedback.",
143 "The choice is yours. ",
144 "\n\n\n\n",
145 "Enter your request ( n,p or j ) : ", 0 } ;
146
147 char *P9[] {
148 "\n\n   The example below shows a typical display of the type of information",
149 "you will see when you request feedback for a given attribute : ",
150 "\n ",
151 "                            8. Size \n",
152 "                             1  1.2 1.4 1.6 1.8  2   4   6   8   10   12 Value",
153 "                            +---+---+---+---+---+---+---+---+---+---+-- Ratio",
154 "                         1  |  ",
155 "         |                  |  ",
156 "       0 | 5             .8 |              1  1",
157 "         |                  |               1  ",
158 "      ---+---            .6 | ",
159 "         |                  | ",
160 "       0 |14             .4 |                 2  1 1",
161 "         |                  |              1  1  1 ",
162 "                           .2 |                 1  2   1 ",
163 "                            |  ",
164 "          Utility Ratio     | ",
165 "\nDo you want to input data y/n ( h for help ) ? ",
166 "\nEnter your request ( n,p or j )  : ",0 } ;
167
168 char *P10[] {
169 "\n\n\n\n",
170 "EXPLANATION OF THE DISPLAY : \n",
```

```
171 "   The display on the left gives the number of data points found in each",
172 "quadrant. If several points are in a quadrant different from the one you",
173 "selected, you might want to see data in other quadrants or change your",
174 "own quadrant. ",
175 "   The points on the graph at the right show the value and utility ratios",
176 "data. An integer k indicates that k subjects provided the same values. ",
177 "   The entry 2 on the graph at the intersection of the value ratio of 1.75",
178 "and the utility ratio of .4 means that two members of the assessment team",
179 "input the pair (1.75,0.4),( or values which round to that pair ). ",
180 "That input means that the two felt that the subject system was 1.75 times",
181 "larger than the baseline system, and the utility for the SIZE of the ",
182 "subject system is only 0.4 of the utility for the SIZE of the baseline",
183 "system.",
184 "   The prompt below the display asks if you want to update your input,",
185 "y for yes, n for no, h for help.",
186 "You may also request any of the options described in HELP. ",
187 "\nEnter your request ( n,p or j ) :",0 } ;
188
189 char *P1[] {
190 "Should you type h you will see :\n",
191     "            INSTRUCTIONS             ",
192     "            ============             ",
193     "  ",
194     " During a session when you are asked to input data,",
195     " you may answer (watch for capital letters) : ",
196     "  ",
197     "  y - YES, and be prompted for data entry ",
198     "  n - NO,  and be prompted to choose another attribute ",
199     "  a - to get the ATTRIBUTE LIST ",
200     "  q - to change the Quadrant    ",
201     "  h - HELP, to get this list  ",
202     "  I - to restart at the Introduction ",
203     "  E - END of session ",
204     "   ",
```

```
205     "Enter any of the above or c to continue \n ",
206 "Should you type any other character you will get this list to remind ",
207 "you of the valid responses ! ",
208 "\n\n",
209 "\nEnter your request (n,p or j ) : ",0 } ;
210
211 char *P12[] {
212 "\n\n",
213 "   More detailed explanations of the commands (a,q,I,F,c) follow : ",
214 "\na - you will get back the attribute list and be able to select another",
215 "    attribute to work with, without entering any data on this one.",
216 "\nq - will let you look at the data in other quadrants at your choice.",
217 "    You will be prompted to input the quadrant number, the data will be ",
218 "    displayed immediatly in the new format as the response. You can use ",
219 "    q again and pick the old or any other quadrant.",
220 "\nI - will get you into this introductory phase again so you will be able",
221 "    to flip pages and clarify anything you need.",
222 "\nL - will terminate the session and will stop the execution of the program",
223 "\nc - will let you chose a new attribute without using the attribute list.",
224 "\n\n\n\n",
225 "Enter your request (n,p or j ) : ",0 } ;
226
227 char *P13[] {
228 "\n\n\n\n",
229 "   This completes the introductory phase. You may now type j and begin",
230 "with the first iteration of data input or you may go back through parts",
231 "of the introduction (by typing p or n) until you feel that you understand",
232 "the instructions. Remember that you can access the introduction later, ",
233 "if necessary, by typing I . ",
234 "\n\n\n\n\n\n\n\n\n\n\n\n\n",
235 "Enter your request ( n,p or j ) : ", 0 } ;
236
237 int i  ;
238 char q ;
```

```
239 intro()
240 {
241 int i,j ;
242 i = 1 ;
243
244 while(1) {
245 switch(i) {
246      case 1 : j =  0 ; while(P1[j] != 0) printf("%s\n",P1[j++]); break ;
247      case 2 : j =  0 ; while(P2[j] != 0) printf("%s\n",P2[j++]); break ;
248      case 3 : j =  0 ; while(P3[j] != 0) printf("%s\n",P3[j++]); break ;
249      case 4 : j =  0 ; while(P4[j] != 0) printf("%s\n",P4[j++]); break ;
250      case 5 : j =  0 ; while(P5[j] != 0) printf("%s\n",P5[j++]); break ;
251      case 6 : j =  0 ; while(P6[j] != 0) printf("%s\n",P6[j++]); break ;
252      case 7 : j =  0 ; while(P7[j] != 0) printf("%s\n",P7[j++]); break ;
253      case 8 : j =  0 ; while(P8[j] != 0) printf("%s\n",P8[j++]); break ;
254      case 9 : j =  0 ; while(P9[j] != 0) printf("%s\n",P9[j++]); break ;
255      case 10 : j =  0 ; while(P10[j] != 0) printf("%s\n",P10[j++]); break ;
256      case 11 : j =  0 ; while(P11[j] != 0) printf("%s\n",P11[j++]); break ;
257      case 12 : j =  0 ; while(P12[j] != 0) printf("%s\n",P12[j++]); break ;
258      case 13 : j =  0 ; while(P13[j] != 0) printf("%s\n",P13[j++]) ;
259
260      }
261
262      g = getchar(); getchar();
263      if(g == 'j') return ;
264      if(g == 'p') i = i - 1 ;
265      else i = i + 1 ;
266      if(i == 0) i=1;
267      if(i == 14) i=13;
268      }
269 return ;
270 }
```

```
 1
 2 //  This separatly compiled file contains the subroutins that
 3 //  displays the four different quadrants.
 4 //  Its grat.o modul needs to loaded with the program 'tt'.
 5
 6 extern char d[16][45],S[40][32];
 7 extern int atn,q[][4];
 8
 9 q1(v1, u1)
10     float v1, u1;
11 {
12     int du, dv;
13     du = dv = 0;
14     if(v1 >= 1 && v1 <= 2)
15         dv = 20*(v1-1.0)+0.00001;
16     else if(v1 >= 2 && v1 <= 12.0)
17         dv = 2*(v1-2.0)+20.00001;
18     else if(v1 > 12)
19         dv = 43;
20     else
21         return;
22     if(u1 >= 1 && u1 <= 2)
23         du = 10*(u1-1.0)+.00001;
24     else if(u1 >= 2 && u1 <= 12.0)
25         du = (u1-2.0)/2+10.00001;
26     else if(u1 > 12)
27         du = 15;
28     else
29         return;
30
31     if(d[du][dv] == ' ')
32         d[du][dv] = '1';
33     else
34         d[du][dv] = d[du][dv]+'\001';
```

```
35
36      return;
37  }
38
39  a2(v1, u1)
40      float v1, u1;
41  {
42      int du, dv;
43      du = dv = 0;
44      if(v1 <= 1 && v1 >= 0.1)
45          dv = 20*v1+23.00001;
46      else if(v1 < .1)
47          dv = 23;
48      else
49          return;
50      if(u1 >= 1 && u1 <= 2)
51          du = 10*(u1-1.0)+.00001;
52      else if(u1 >= 2 && u1 <= 12.0)
53          du = (u1-2.0)/2+10.00001;
54      else if(u1 > 12)
55          du = 15;
56      else
57          return;
58      if(a[du][dv] == ' ')
59          d[du][dv] = '1';
60      else
61          d[du][dv] = (a[du][dv]+'\001');
62      return;
63  }
64
65  a3(v1, u1)
66      float v1, u1;
67  {
68      int du, dv;
```

```
 69      du = dv = 0;
 70      if(v1 <= 1 && v1 >= 0.1)
 71          dv = 20*v1+23.00001;
 72      else if(v1 < .1)
 73          dv = 23;
 74      else
 75          return;
 76      if(u1 <= 1 && u1 >= 0.1)
 77          du = 10*u1+0.00001;
 78      else if(u1 < 0.1)
 79          du = 0;
 80      else
 81          return;
 82      if(d[du][dv] == ' ')
 83          d[du][dv] = '1';
 84      else
 85          d[du][dv] = (d[du][dv]+'\001');
 86      return;
 87  }
 88
 89  g4(v1, u1)
 90      float v1, u1;
 91  {
 92      int du, dv;
 93      du = dv = 0;
 94      if(v1 >= 1 && v1 <= 2)
 95          dv = 20*(v1-1.0)+0.00001;
 96      else if(v1 >= 2 && v1 <= 12.0)
 97          dv = 2*(v1-2.0)+20.00001;
 98      else if(v1 > 12)
 99          dv = 43;
100      else
101          return;
102      if(u1 <= 1 && u1 >= 0.1)
```

```
103         du = 10*ul+0.0001;
104     else if(ul < 0.1)
105         du = 0;
106     else
107         return;
108     if(d[du][dv] == ' ')
109         d[du][dv] = '1';
110     else
111         d[du][dv] = d[du][dv]+'\001';
112
113     return;
114 }
115
116 graf1() {
117     int i;
118     printf("\n\t\t\t\t\t %2d %s \n\n",atn , S[atn]);
119     printf("\t    Utility     |%s\n", d[15]);
120     printf("\t                10 |%s\n", d[14]);
121     printf("\t                 8 |%s\n", d[13]);
122     printf("\t                 6 |%s\n", d[12]);
123     printf("\t                 4 |%s\n", d[11]);
124     printf("\t    |            2 |%s\n", d[10]);
125     printf("\t%2d | %2d          |%s\n", q[atn][1], q[atn][0], d[9]);
126     printf("\t    |          1.8|%s\n", d[8]);
127     printf("\t---+---       -|%s\n", d[7]);
128     printf("\t    |          1.6|%s\n", d[6]);
129     printf("\t%2d | %2d          |%s\n", q[atn][2], q[atn][3], d[5]);
130     printf("\t    |          1.4|%s\n", d[4]);
131     printf("\t                  |%s\n", d[3]);
132     printf("\t                1.2|%s\n", d[2]);
133     printf("\t                  |%s\n", d[1]);
134     printf("\t                 1 |%s\n", d[0]);
135     printf("\t\t\t       |+---+---+---+---+---+---+---+---+---+---+---\n");
136     printf("\t\t\t        1  1.2 1.4 1.6 1.8  2   4   6   8  10  12 ");
```

```
        printf(" Value \n");

        return;
}

grat2() {
        int i;
        printf("\n\t\t\t\t\t %2d %s \n\n", atn, S[atn]);
        printf("\t                    %s| Utility \n", d[15]);
        printf("\t                    %s|10 \n", d[14]);
        printf("\t                    %s| 8 \n", d[13]);
        printf("\t                    %s| 6 \n", d[12]);
        printf("\t    |               %s| 4 \n", d[12]);
        printf("\t%2d | %2d             %s| 2 \n", q[atn][1], q[atn][0], d[10]);
        printf("\t    |               %s|   \n", d[9]);
        printf("\t---+---             %s|1.8 \n", d[8]);
        printf("\t    |               %s|   \n", d[7]);
        printf("\t%2d | %2d             %s|1.6 \n", q[atn][2], q[atn][3], d[6]);
        printf("\t    |               %s|   \n", d[5]);
        printf("\t                    %s|1.4 \n", d[4]);
        printf("\t                    %s|   \n", d[3]);
        printf("\t                    %s|1.2 \n", d[2]);
        printf("\t                    %s|   \n", d[1]);
        printf("\t                    %s| 1\n", d[0]);
        printf("\t                                            --+-+-+-+-+-+-+-+-+-+|\n");
        printf("\t                                        Value    .2   .4   .6   .8    1 \n");
        return;
}

grat3() {
        int i;
        printf("\n\n\n\n\n\n\t\t\t\t\t %2d %s \n\n", atn, S[atn]);
        printf("\t                                        value    .2   .4   .6   .8    1 \n");
        printf("\t                                            --+-+-+-+-+-+-+-+-+-+|\n");
```

```
171     printf("\t           %s| 1 \n", d[10]);
172     printf("\t    |      %s|   \n", d[9]);
173     printf("\t%2d | %2d   %s|.8 \n", q[atn][1], q[atn][0], d[8]);
174     printf("\t    |      %s|   \n", d[7]);
175     printf("\t---+---    %s|.6 \n", d[6]);
176     printf("\t    |      %s|   \n", d[5]);
177     printf("\t%2d | %2d   %s|.4 \n", q[atn][2], q[atn][3], d[4]);
178     printf("\t    |      %s|   \n", d[3]);
179     printf("\t           %s|.2 \n", d[2]);
180     printf("\t           %s|   \n", d[1]);
181     printf("\t           %s| Utility   \n", d[0]);
182     return;
183 }
184
185 graf4() {
186     int i;
187     printf("\n\n\n\n\n\n\t\t\t\t\t %2d %s \n\n", atn, S[atn]);
188     printf("\t\t\t              1  1.2 1.4 1.6 1.8  2   4   6   8  10  12 ");
189     printf(" Value \n");
190     printf("\t\t\t           |+---+---+---+---+---+---+---+---+---+---+---\n");
191     printf("\t                1 |%s\n", d[10]);
192     printf("\t    |             |%s\n", d[9]);
193     printf("\t %2d | %2d          .8|%s\n", q[atn][1], q[atn][0], d[8]);
194     printf("\t    |             |%s\n", d[7]);
195     printf("\t ---+---         .6|%s\n", d[6]);
196     printf("\t    |             |%s\n", d[5]);
197     printf("\t %2d | %2d          .4|%s\n", q[atn][2], q[atn][3], d[4]);
198     printf("\t    |             |%s\n", d[3]);
199     printf("\t                .2|%s\n", d[2]);
200     printf("\t                  |%s\n", d[1]);
201     printf("\t          Utility   |%s\n", d[0]);
202
203     return;
204 }
```

```
 1
 2 //       GROUP DECISION MAKING WITH FEEDBACK
 3 //       ===================================
 4 //
 5 //       The analysis program.
 6 //       ---------------------
 7 //
 8 //
 9 #define NUSERS 16   // maximum number of users + 1
10 # define NATTR  28   // maximum number of attributes + 1
11 #define BNKSIZE 16   // size of a data record in bytes
12
13 //  structure data defines the basic data record. the fields are:
14 //
15 //  dm    - display quadrant at data entry time
16 //  pd    - pointer to the privious data record by the same user
17 //          on the same attribute
18 //  usn   - the user unique ID number
19 //  atrn  - the attribute number
20 //  vsl   - the value ratio entered
21 //  usl   - the utility ratio entered
22 //  time[2] - system time at data entry
23
24 struct data {
25      int dm, pd, usn, atrn, vsl, usl, time[2];
26 };
27
28 //  the structure attr holds the constant size information about
29 //  the users and the attributes.
30 //
31 //  usr    - pointer to the last data record entered by a given user
32 //           about a given attribute
33 //  count  - tally of the number of changes to a given attribute
34 //           since last updated by a given user
```

```
35 //  prof  - the self-rated proficiency for each user-attribute pair
36 //  quadr - the quadrant the user choses to see the data in
37
38 struct attr {
39      int usr[NUSERS], count[NUSERS], prof[NUSERS], quadr[NUSERS];
40 };
41
42 struct data bank[2048];
43 struct attr att[NATTR];
44 struct data *B;
45
46 char d[16][45];
47 int a[NATTR][4];
48 int geti();
49 double getf();
50 char getchar();
51
52 char *c1 "bnk";
53 char *c2 "atb";
54 char *cf "flag";
55 char *ct "atlist";
56 int M -1;
57 int un, atn, atsize, natt, fsize, fda, fdb;
58 char c;
59 char S[40][32]; // Will hold the attribute list
60                 // created by the program 'atlst'
61
62 //  The main program enables the monitor to monitor the progress of
63 //  the working session.
64 //  The program permits the monitor to do the following :
65 //
66 //      a. Data records in numerical format can be displayed for any
67 //         any single user or all users and any single attribute or
68 //         all attributes.
```

```
69 //      b. The data in graphical presentation (as the users see it)
70 //         can be displayed for any user and any attribute, as
71 //         they are at any selected time present or past.
72 //      c. The data in graphical presentation but from any proficiency and
73 //         up.
74 //
75 main() {
76      int i, j, k, l, fdt;
77      atsize = 8*NATTR*NUSERS;
78      M = -1;
79      B = &bank;
80      fda = open(c2, 2);
81      fdb = open(c1, 2);
82      fdf = open(cf, 2);
83      fdt = open(ct, 2);
84      seek(fdt, 0, 0);
85      read(fdt, &natt, 2);
86      read(fdt, S, 1280);
87
88      while(1) {
89          seek(fdb, 0, 0);
90          read(fdb, bank, 32767);
91          seek(fda, 0, 0);
92          read(fda, att, atsize);
93
94          printf("\n\nEnter user # , 99 for all , 66 for time cuts ");
95          printf(", NEGATIVE to quit. \n");
96          i = geti();
97          if(i < 0)
98              exit();
99          if(i == 66) {
100             outdata();
101             continue;
102         }
```

```
103             un = i;
104             printf("Enter attribute # , 99 for all \n");
105             atn = geti();
106             if(un != 99 && atn != 99) {
107                 stat(un, atn);
108                 continue;
109             }
110             if(un != 99 && atn == 99) {
111                 for(j = 0; j < natt; j++)
112                     stat(un, j);
113                 continue;
114             }
115             if(un == 99 && atn != 99) {
116                 for(i = 0; i < NUSERS; i++)
117                     stat(i, atn);
118                 continue;
119             }
120             for(i = 0; i < NUSERS; i++)
121                 for(j = 0; j < natt; j++)
122
123                     stat(i, j);
124
125         }
126 }
127
128 //  The stat routine writes the record of data of any given
129 //  user - attribute pair.
130 //
131
132 stat(i, j)
133     int i, j;
134 {
135     struct data *p;
136     int ip, t0, t1, od, of;
```

```
137     float fv1, fu1;
138
139     ip = att[j].usr[i];
140     p = B+ip;
141
142     while(ip != M) {
143         fv1 = (p->vs1)/1000.0;
144         fu1 = (p->us1)/1000.0;
145         t0 = p->time[0];
146         t1 = p->time[1];
147         ip = p->pd;
148         qd = p->dm;
149         pf = att[j].prof[i];
150         p = B+ip;
151         printf("un= %2d  atn= %2d  prof= %1d  quad= %1d  valr= %2.2f  ", i, j, pf,p
152         printf("utyr= %2.2f  t0t1= %d %d\n", fu1, t0, t1);
153
154
155     }
156     return;
157 }
158
159 //  The outdata routine get the data and prepare it for graphical presntation
160 //  Performs also the time and proficiency cuts.
161 //
162
163 outdata() {
164     struct data *p;
165     float v1, u1;
166     int i, k, l, dv, du, pr, ttvc[2];
167     unsigned cut, tvec[2];
168     time(ttvc);
169     tvec[0] = ttvc[0];
170     tvec[1] = ttvc[1];
```

```
171     cut = 0;
172
173     v1 = u1 = 1.0;
174
175     for(k = 0; k < 16; k++) {        // Initialize the data display array
176         for(l = 0; l < 44; l++)
177             d[k][l] = ' ';
178         d[k][44] = '\0';
179     }
180     for(k = 0; k < NATTR; k++) // Initialize the quadrant presentation array
181         for(l = 0; l < 4; l++)
182             q[k][l] = 0;
183
184     seek(fda, 0, 0);
185     read(fda, att, atsize); //Read the data bank.
186     seek(fdb, 0, 0);
187     read(fdb, bank, 32767);
188     printf("Enter attribute you want the cut for : ");
189     do
190         atn = geti();
191     while(!(atn >= 1 && atn <= (natt-1)));
192     printf("Enter the quadrant for display : ");
193     do
194         i = geti();
195     while(!(i >= 1 && i <= 4));
196     printf("Enter the proficiency level : ");
197     do
198         pr = geti();
199     while(!(pr >= 1 && pr <= 5));
200     printf("Time now is %d ,enter the time cut : ", tvec[1]);
201     while(cut == 0)
202         cut = geti();
203     for(l = 0; l < NUSERS; l++) {
204         k = att[atn].usr[l];
```

```
205         p = R[k];
206         if(k == M || p->time[1] > cut || att[atn].prof[1] < pr)
207             continue;
208         else
209             v1 = (p->vs1)/1000.0+0.001  // Refloat the ratios .
210         u1 = (p->us1)/1000.0+0.001;
211         if(v1 >= 1 && u1 >= 1)
212             qlatn[0]++;                 // Prepare the quadrant presentation.
213         if(v1 < 1 && u1 >= 1)
214             qlatn[1]++;
215         if(v1 < 1 && u1 < 1)
216             qlatn[2]++;
217         if(v1 >= 1 && u1 < 1)
218             qlatn[3]++;
219         switch(i) {      // Call the proper data display preparation routine.
220         case 1:
221             q1(v1, u1);
222             break;
223         case 2:
224             q2(v1, u1);
225             break;
226         case 3:
227             q3(v1, u1);
228             break;
229         case 4:
230             q4(v1, u1);
231         }
232     }
233     switch(i) {      // Call the proper graphical display routine.
234     case 1:
235         graf1();
236         break;
237     case 2:
238         graf2();
```

```
239             break;
240         case 3:
241             graf3();
242             break;
243         case 4:
244             graf4();
245         }
246         return;
247 }
248
```

```
 1
 2 //  This program lets the manager to input the attribute list into
 3 //  a file from where it will be used by the user's program 'tt'
 4 //  and the analysis program 'an'. This avoids doing the same by
 5 //  editing and recompiling the same programs.
 6
 7 char *ca "atlist" ;
 8
 9 main()
10   {
11     int i,j,k,l,fda ;
12     char S[40][32] ;
13     char c,inst ;
14     c = 'c' ;
15     fda = open(ca,2) ;
16     seek(fda,0,0) ;
17     read(fda,&i,2) ;
18     read(fda,S,1280) ;
19     printf("This program will let you update an old or start a new \n") ;
20     printf("attribute list depending on whether you type u or n   \n") ;
21     printf("Your instruction is  \n");
22     inst = getchar() ;
23     getchar() ;
24     if(inst == 'n') {
25     for(j=0;j<40;j++) {for(k=0;k<31;k++) S[j][k] = ' ' ; S[j][31] = '\0' ; }
26     i = 0 ;
27     while(c != '@' )
28       {
29         j = 0 ;
30         printf("Enter attribute # %3d (@ when you are done) \n",++i) ;
31         while((c = getchar()) != '\n') { S[i][j++] = c ;
32                                          if(j == 31) break ; }
33         c = S[i][j-1] ;
34       }
```

```
35      }
36      else {
37          while(1) {
38          j = 0 ;
39          printf("Enter the # of the attribute you want to update, \n") ;
40          printf("0 when you are done. \n") ;
41          do l = geti() ;
42          while(!(l >= 0 && l <= 38) ) ;
43          if(l == 0) break ;
44          for(k=0;k<31;k++) S[l][k] = ' ' ; S[l][31] = '\0' ;
45          printf("Enter attribute # %3d \n",l) ;
46          while((c = getchar()) != '\n') { S[l][j++] = c ;
47                                           if(j == 31) break ; }
48          }
49      }
50
51      for(j=1;j<=i/2;j++)
52        printf("%2d %s     %2d %s \n",j,S[j],i/2+j,S[i/2+j]) ;
53      seek(fda,0,0) ;
54      write(fda,&i,2) ; // put the number of attributes at the begining
55      write(fda,S,1280) ;
56
57  }
```

```
35                                                              att[i].count[j]= 0 ; }
36          unlink(c1)  ;
37          creat(c1,2) ;
38          creat(c2,2) ;
39          creat(cf,2) ;
40          fda = open(c2,2) ;
41          fdb = open(c1,2) ;
42          fdf = open(cf,2) ;
43          chmod(c1,07777) ;
44          chmod(c2,07777) ;
45          chmod(cf,07777) ;
46          write(fda,att,atsize) ;
47          write(fdf,&fsize,2) ;
48      printf("\n\n\n\n\t\t\tDATA BANK CLEARED !!! \n\n") ;
49      return ;
50                  }
51      printf("\n\n\n\t\t\tNO CLEAR !!! \n\n") ;
52      return ;
53  }
```

```
 1
 2 //  This program will clear and initialize the database files.
 3 //
 4 //  A message to the manager is given to assert that he realy
 5 //  wants to do that.
 6
 7 #define NUSERS 16
 8 #define NATTR  28
 9
10 struct  data {
11      int dm,pd,usn,atrn,vsl,usl,time[2] ;
12      } ;
13 struct data bank[2048] ;
14
15 struct attr
16    {
17      int usr[NUSERS],count[NUSERS],prof[NUSERS],quadr[NUSERS] ;
18    } ;
19 struct attr att[NATTR] ;
20 char *c1 "bnk" ;
21 char *c2 "atb" ;
22 char *cf "flag" ;
23 int fsize 0 ;
24
25 main()
26    {
27      int i,j,fda,fdb,fdf,atsize ;
28      char c ;
29      atsize = NUSERS * NATTR * 8 ;
30      printf("\n\n\n\n\n\nthis program will ERASE the current data bank !!");
31      printf("\n\n\nType Y if you want to proceed, q to quit \n\n\n\n\n\n") ;
32      c = getchar() ;
33      if(c == 'Y') {
34          for(i=0,i<NATTR;i++) for(j=0;j<NUSERS;j++) {att[i].usr[j] = -1 ;
```

```
 1
 2 //  This program serves as the database administrator and its job
 3 //  is to give write permition to one user's program at a time.
 4 #define NUSERS 16
 5 char *cr "request" ;
 6 char *ck "ticket" ;
 7 char *cc "count" ;
 8 char *cp "compare" ;
 9 char *ct "T" ;
10 int t 1 ;
11 int rqst[NUSERS] 0 ;
12 int tkt[NUSERS]  0 ;
13 int comp[NUSERS] 0 ;
14 int Z[1] 1 ;
15 int zero 0 ;
16 char a ;
17
18 main()
19   {
20     int i,fdr,fdk,fdc,fdp,fdt,N ;
21     char a ;
22     N = 2 * NUSERS ;
23     fdr = open(cr,2) ;
24     fdk = open(ck,2) ;
25     fdc = open(cc,2) ;
26     fdp = open(cp,2) ;
27     fdt = open(ct,2) ;
28     printf(" IS IT A NEW DATA BANK ?   Y / N \n\n") ;
29     a = getchar() ;
30     if(a == 'Y' ) {
31         write(fdc,Z,2) ;     //this is a new database, the files are initialized
32         write(fdk,tkt,N) ;
33         write(fdp,comp,N) ;
34         write(fdt,&t,2) ;
```

```
35          write(fdr,rqst,N) ; }
36          else {
37          read(fdk,tkt,N);     // this is an old file, the arrays are
38          read(fdr,rqst,N) ;   // initialized to the existing files
39          read(fdp,comp,N) ;
40          read(fdt,&t,2) ; }
41
42      seek(fdr,0,0) ;
43      write(fdr,&zero,2) ;
44      printf("\n\n\n\t\t\tBOX IS RUNNING !!\n\n\n") ;
45
46
47      while(1) {
48          seek(fdr,0,0) ;
49          read(fdr,rqst,N) ;
50          if(rqst[0] == -1) { printf("\n\n\t\tBOX ENDS\n"); exit(); }
51          for(i=0;i<NUSERS;i++) {
52              if(rqst[i] < comp[i]) {tkt[i] = ttt ; comp[i] = rqst[i] ; }
53              if(rqst[i] > comp[i]) {seek(fdc,0,0) ;
54                      write(fdc,&rqst[i],2) ; comp[i] = rqst[i] ; }
55            }
56      seek(fdk,0,0) ;
57      write(fdk,tkt,N) ;
58      seek(fdp,0,0) ;
59      write(fdp,comp,N) ;
60      seek(fdt,0,0) ;
61      write(fdt,&t,2) ;
62      sleep(5) ;
63      }
64  }
65
```

```
 1
 2 //  This program will link and execute the Group Decision Making
 3 //  program 'tt' .
 4
 5 main()
 6 {
 7      int i;
 8      if((i = fork()) == 0){
 9          execl("/usr/tamir/tt","/usr/tamir/tt",0);
10          exit();
11      }
12      wait(&i);
13 }
```

```
 1
 2 //  This little program sets the -1 flag in request[0] as a sign
 3 //  to tbox to end its work.
 4 //  The program is needed since usualy tbox will run in the background.
 5
 6 char *cr "request" ;
 7 int Z -1 ;
 8
 9 main() {
10      int fdr ;
11      fdr = open(cr,2) ;
12      seek(fdr,0,0) ;
13      write(fdr,&Z,2) ;
14 }
```

```
 1
 2 // This program will convert the database in the files 'bnk' and
 3 //  'atb', which are in PDP11 integer format, to long integers
 4 //  32 bits long, so that the database is IBM compatible
 5
 6 int i,j,fsize,nb,rs,k ;
 7 char *cb "bnk" ;
 8 char *ca "atb" ;
 9 char *lb "lbnk" ;
10 char *la "latb" ;
11 int I[16384] ;
12 long L[2501 ;
13
14 main()
15   {
16     int fda,fdb,fla,flb ;
17
18     unlink(lb) ;
19     unlink(la) ;
20     creat(lb,2) ;
21     creat(la,2) ;
22     chmod(la,07777) ;
23     chmod(lb,07777) ;
24     fda = open(ca,2) ;
25     fdb = open(cb,2) ;
26     fla = open(la,2) ;
27     flb = open(lb,2) ;
28
29
30 //  The data is converted and written in blocks of 512 bytes, which
31 // is the efficient size for the UNIX i/o.
32
33     seek(fdb,0,0) ;
34     fsize = read(fdb,I,16384) ;
```

```
35      nb = fsize / 256 ;
36      rs = (fsize%256) / 2 ;
37      for(j=0;j<nb;j++) {
38          for(k=0;k<128;k++)   L[k] = I[j * 128 + k] ;
39          write(flb,L,512) ; }
40      for(k=0;k<rs;k++)   L[k] = I[nb * 128 + k] ;
41      write(flb,L,rs * 4) ;
42
43      seek(fda,0,0) ;
44      fsize = read(fda,I,16384) ;
45      nb = fsize / 256 ;
46      rs = (fsize%256) / 2 ;
47      for(j=0;j<nb;j++) {
48          for(k=0;k<128;k++)   L[k] = I[j * 128 + k] ;
49          write(fla,L,512) ; }
50      for(k=0;k<rs;k++)   L[k] = I[nb * 128 + k] ;
51      write(fla,L,rs * 4) ;
52
53      printf("\t\tDATA HAS BEEN CONVERTED TO IBM COMPATIBLE FORMAT.\n\n\n");
54  }
```

extpdsk exec

```
FILEDEF 2 TAP2 RECFM F LRECL 32 BLKSIZE 32
FILEDEF 3 DSK ATTR FILE RECFM F LRECL 32 BLKSIZE 32
FILEDEF 4 DSK BANK FILE RECFM F LRECL 32 BLKSIZE 32
LOAD TAPDSK (XEQ)
```

tapdsk fortran

```
C
C    THIS PROGRAM WILL READ TWO FILES FROM A TAPE AND
C    WRITES THEM INTO A DISK ON THE CP/CMS FILE SYSTEM
C
C
      INTEGER*4 A(8)
      I = 0
    1 READ(2,100,END=500) A
  100 FORMAT (8A4)
      WRITE(3,200) A
  200 FORMAT(8A4)
      I = I + 1
      GO TO 1
  500 END FILE 2
      WRITE(6,505) I
  505 FORMAT(' END OF FILE ATTR.',I8,' RECORDS.')
      I = 0
    2 READ(2,100,END=600) A
      WRITE(4,200) A
      I = I + 1
      GO TO 2
  600 END FILE 2
      WRITE(6,605) I
  605 FORMAT(' END OF FILE BANK.',I8,' RECORDS.')
      STOP
      END
```

LIST OF REFERENCES

1. Von Neumann, J. and O. Morgenstern, Theory of Games and Economic Behavior, 2nd ed. Princeton University Press, Princeton, N.J., 1947.

2. Keeney, R.L. and Raiffa, H, Decisions with Multiple Objectives. John Wiley & Sons, 1976.

3. Zimmermann, H., Fuzzy Set Theory. Class notes and handout, Naval Postgraduate Schoo, Monterey, CA., 1979.

4. Raiffa, H., Decision Analysis, Introductory Lectures on Choices Under Uncertainty. Addison-Wesley, 1968.

5. Fishburn, P.C., Decision and Value Theory. John Wiley & Sons, 1964.

6. Barr, D.R. and Richards, F.R., Utility Assessment Methodology, A System Exploration. Preliminary Report, 1978.

7. Knuth, D.E., The Art of Computer Programming, Vol. 1, 2nd Ed. Addison-Wesley Publishing Company, 1973.

8. Date, C.J., An Introduction to Data Base Systems, 2nd Ed. Addison-Wesley Publishing Company, 1977.

9. Kernighan, B.W. and Ritchie, D.M., The C Programming Language, Englewood Cliffs, N.J.: Prentice-Hall, 1978.

10. The Bell System Technical Journal, July-Aug 1978, Vol. 57, No. 6, Part 2.

11. Reed, D.P., and K.K. Rajendra, "Synchronization with Eventcount and Sequencers," Communication of the ACM. Feb. 1979, Vol. 22, No. 2.

12. Thompson, K., Ritchie, D.M., Unix Programmers Manual, 6th Ed. Bell Telephone Laboratories, Inc., 1975.

13. Sackman, S., "Delphi Assessment: Expert Opinion, Forecasting, and Group Process," Rand Corporation Report R-1283-PR, April 1974.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Documentation Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, CA 93940	2
3.	Department Chairman, Code 55 Department of Operations Research Naval Postgraduate School Monterey, CA 93940	1
4.	Department Chairman, Code 52 Department of Computer Science Naval Postgraduate School Monterey, CA 93940	1
5.	Assoc. Professor F.R. Richards, Code 55Rh Naval Postgraduate School Monterey, CA 93940	5
6.	Professor G.H. Bradley, Code 52 Naval Postgraduate School Monterey, CA 93940	1
7.	Lt. Col. R.J. Roland, Code 52Rl Naval Postgraduate School Monterey, CA 93940	2
8.	Major Amnon Tamir 15 Rabina St. Neve Avivim Tel-Aviv. ISRAEL	2
9.	Air Attache, Embassy of ISRAEL 1621 22nd St. N.W. Washington, D.C. 20008	3

Thesis
T1347 Tamir
c.1 Group decision making
with feedback.